การพัฒนาโปรแกรมบน iPad เพื่อใช้ในงานสนับสนุนการตลาด

Development of Marketing Support Application on iPad Platform

นายศิริชัย จันทร์นิตย์

โครงงานสหกิจศึกษานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีไทย – ญี่ปุ่น
พ.ศ. 2554

โปรแกรมพัฒนาและสนับสนุนการตลาด CPAll บน iPad

ในแผนกวิจัยและพัฒนาผลิตภัณฑ์ บริษัท โกซอฟท์ ประเทศไทยจำกัด

Application develoment and markerting support CPALL on iPad

Reach and development gosoft (Thailand) Co.,Ltd.

นายศิริชัยจันทร์นิตย์

โครงงานสหกิจศึกษานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาตรี สาขาวิชาคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีไทย – ญี่ปุ่น

พ.ศ. 2554

คณะกรรมการสอบ

.....ประธานกรรมการสอบ

(อาจารย์ปริวัตร คงกำเนิด)

.....กรรมการสอบและอาจารย์ที่ปรึกษา

(อาจารย์วรวุฒิจิตขจรวานิช)

.....กรรมการ

(อาจารย์ ดร.วิมล แสนอุ้ม)

บทสรุป

ชื่อโครงงาน	การพัฒนาโปรแกรมบน iPad เพื่อใช้ในงานสนับสนุนการตลาดพ.ศ. 2554 Development of Marketing Support Application on iPad Platform ค.ศ. 2011
ผู้เขียน	นายศิริชัยจันทร์นิตย์
คณะวิชา	วิศวกรรมศาสตร์สาขาวิชาคอมพิวเตอร์
อาจารย์ที่ปรึกษา	อาจารย์วรวุฒิ จิตขจรวานิช
พนักงานที่ปรึกษา	คุณอรวรา โรจนวาศ (Head, Business Development) นางอรุณวรรณ นาจารย์(Leader, Business Development) นายเฉลิมชัย สถานุรักษ์วงศ์ (Software Engineering)
ชื่อบริษัท	บริษัท โกซอฟท์ ประเทศไทยจำกัด
ประเภทธุรกิจ / สินค้า	Application Development, System Intergrater, Professional Service, DataCenter, Call Services

งานที่ปฏิบัติ

เมื่อข้าพเจ้าได้เข้ามาปฏิบัติงานที่บริษัทโกซอฟท์(ประเทศไทย)จำกัด ได้รับมอบหมายงานเขียนซอฟท์แวร์ชื่อ StoreHereเป็นโปรแกรมประยุกค์ที่ใช้ในการสนับสนุนการขายของข้าวตราฉัตรโดยโปรแกรมประยุกค์ตัวนี้จะมีการแสดง ที่ตั้งของร้านต่างๆ ทั้งลูกค้า และร้านค้า บนแผนที่ สามารถทำการเปรียบเทียบราคาสินค้า สั่งของ แสดงเส้นทาง และ ถ่ายรูปได้

ผลที่ได้รับจากการดำเนินงานและประโยชน์ที่ได้รับ

จากการได้ทำงานสหกิจศึกษาเป็นเวลา 4 เดือนนั้น ได้ทำการศึกษาภาษาใหม่ๆ ในการเขียนโปรแกรม และประสบการณ์ในการทำงานจริง ร่วมกับบุคคลอื่นๆ

กิตติกรรมประกาศ

ในช่วยระยะเวลา 4 เดือนที่ข้ามเจ้าได้ปฏิบัติงานสหกิจศึกษาณ บริษัทโกซอฟท์(ประเทศไทย)จำกัดนั้น ข้าพเจ้าได้รับการดูแลและคำปรึกษาจากบุคคลหลายท่าน ทำให้การทำงานเป็นไปได้ด้วยดี และได้รับประสบการณ์ที่ไม่สามารถหาได้ในห้องเรียน

การทำงานครั้งนี้จะสำเร็จไปไม่ได้ถ้าขาดการดูแลจากทุกคน ขอขอบคุณคุณอรวรา โรจนวาศ (Head Business Development), คุณอรุณวัฒ นาจารย์ (Leader Business Development) ผู้ที่คอยดูให้คำแนะนำเรื่องต่างๆ ทั้งเรื่องการปฏิบัติตัวในองค์กร และการใช้เครื่องมือในสำนักงาน, คุณเฉลิมชัย สถานุรักษ์วงศ์ (Software Engineer) เป็นผู้ให้คำแนะนำในการเขียน โปรแกรมพร้อมทั้งสอนเรื่องต่างๆ ให้ ทั้งหมดนี้ทำให้ข้าพเจ้าได้รับประสบการณ์ดีๆ กลับไปมากมายจึงขอขอบคุณบริษัทโกซอฟท์ที่ให้โอกาสข้าพเจ้าได้มาปฏิบัติสหกิจศึกษา ณ ที่นี้ด้วย

ศิริชัย จันทร์นิตย์

ผู้จัดทำรายงาน

สารบัญ

ประวัติผู้วิจัย 87

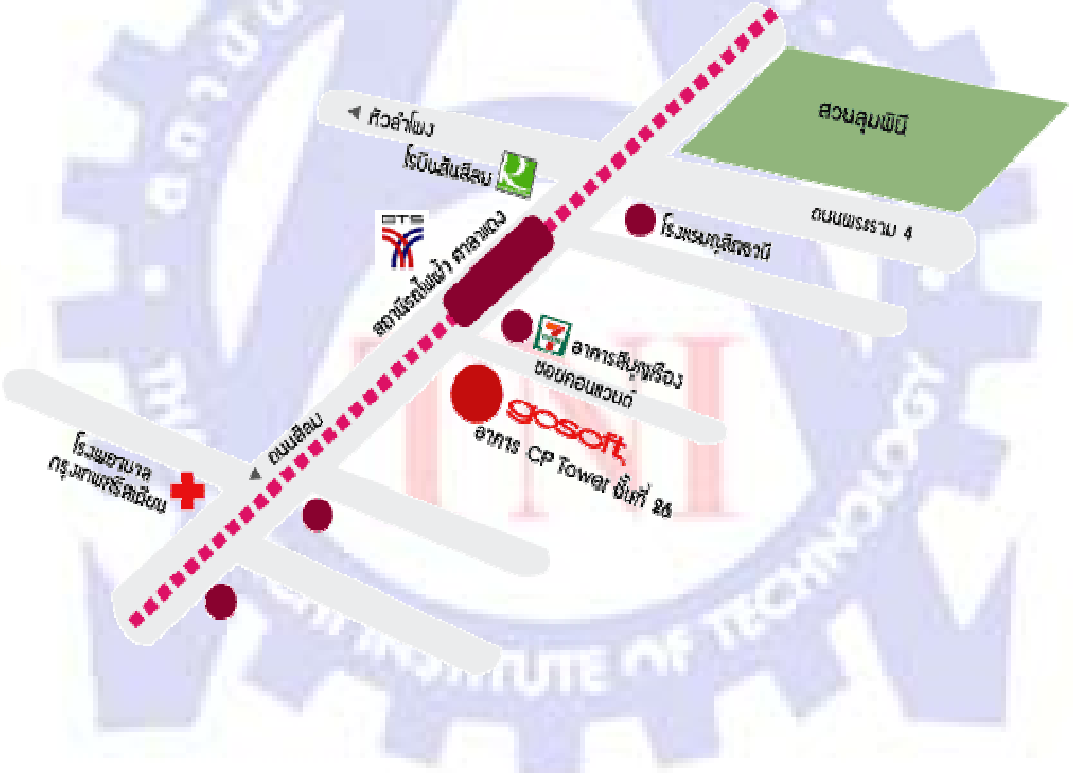
รายการตาราง

รายการรูปประกอบ

บทที่ 1
บทนำ

1.1 ชื่อและที่ตั้งของสถานประกอบการ

ชื่อหน่วยงาน	โกซอฟท์ ประเทศไทย จำกัด (gosoft Thailand Co.,Ltd)
ที่ตั้ง	เลขที่ 313 อาคาร ซีพีทาวน์เวอร์ ชั้น 26 ห้อง 2104 ถนนสีลม แขวงสุริยวงศ์ เขตบารักกรุงเทพมหานคร 10500
โทรศัพท์	(662) 677-9451
แฟกซ์	(662) 677-9400
เว็บไซต์	http://www.gosoft.co.th



ภาพที่1.1แสดงสถานที่ตั้ง บริษัทโกซอฟท์ (ประเทศไทย) จำกัด

1.2 ลักษณะธุรกิจของสถานประกอบการ หรือการให้บริการหลักขององค์กร

1.2.1 บทย่อ

บริษัทโกซอฟท์ (ประเทศไทย) จำกัด หรือ gosoft (Thailand) Co.,Ltd. เป็นบริษัทฯในกลุ่มธุรกิจของบริษัท ซี.พี. เซเว่นอีเลฟเว่น จำกัด (มหาชน) ซึ่งปัจจุบันคือบริษัท ซีพี ออลล์ จำกัด (มหาชน) จัดตั้งเป็นบริษัทเมื่อวันที่ 1 มกราคม 2546 ก่อตั้งขึ้นด้วยการนำองค์กรด้าน IT ที่มีประสบการณ์หลายสายงานภายในกลุ่มให้เป็นหนึ่งเดียวกันจึงทำให้บริษัทเป็นแหล่งความรู้ (Know How) และเทคโนโลยีที่หลากหลายด้าน Hardware , Software และ Solution ต่าง ๆ

วัตถุประสงค์ของบริษัทฯคือเป็นบริษัทชั้นนำในการให้คำปรึกษาและบริการด้านสารสนเทศ (Information System) ในประเทศไทย ซึ่งให้บริการครอบคลุมถึงการพัฒนาโปรแกรมใช้งายภายในธุรกิจ (In-house Application Development) , ติดตั้งระบบโปรแกรมชั้นนำ (System Integrator) , ระบบเครือข่าย (Networking) , ระบบรักษาความปลอดภัยข้อมูล (Security System) และอื่น ๆ

นอกจากนี้บริษัท ยังมีศูนย์ข้อมูล Data Center และศูนย์ Call Service ให้บริการแก่ลูกค้าด้วย ส่วนของ Data Center เป็นศูนย์ที่มีมาตรฐานระดับสูง Tier IV ซึ่งมีระบบสำรองในจุดที่มีความเสี่ยงต่าง ๆไว้เพื่อให้ความมั่นใจถึงความปลอดภัย และความเสถียร ของข้อมูล พื้นที่ตั้งของศูนย์แยกอย่างอิสระจากพื้นที่ประกอบธุรกิจ (Isolated Area) เพื่อลดความเสี่ยงในการเกิดเหตุการณ์ต่าง ๆในส่วนของสำนักงาน Call Service เป็นศูนย์บริการลูกค้าในเรื่องต่าง ๆ(Helpdesk), บริการลูกค้าสัมพันธ์ (CRM), แจ้งเรื่อง (Call Dispatch) และอื่น ๆให้บริการลักษณะ 7 วัน หรือ 24 ชั่วโมง

1.2.2 วิสัยทัศน์ของบริษัท

"เราให้บริการเทคโนโลยีสารสนเทศ และ คอนแทคเซ็นเตอร์ อย่างมีคุณภาพ"

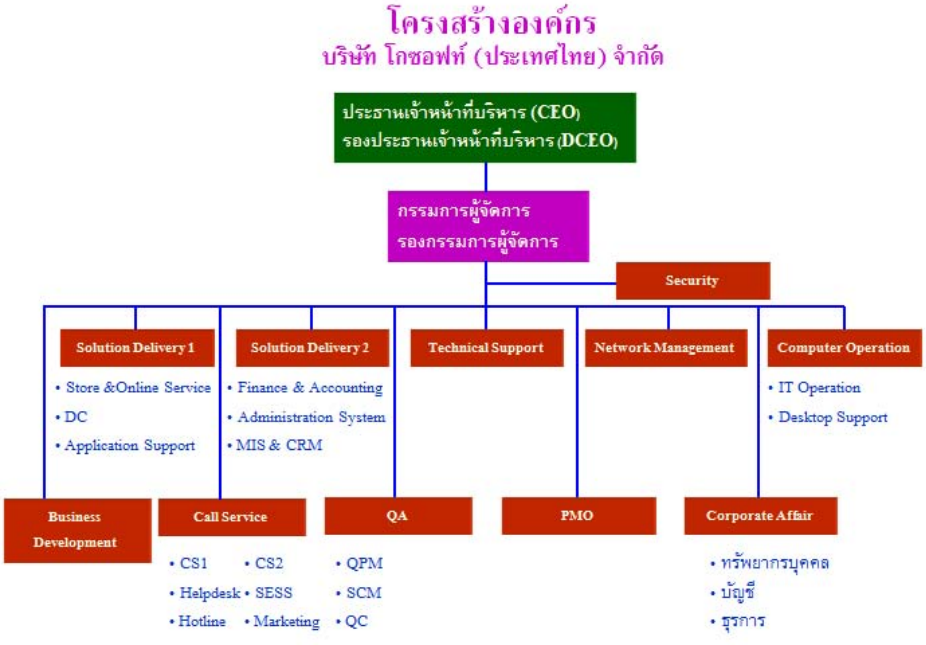
1.2.3 ภาระกิจของบริษัท

- สร้างผลการดำเนินงานให้เติบโตและมีผลกำไรที่ยั่งยืน
- ส่งมอบและให้บริการระบบงานเทคโนโลยีสารสนเทศ และ คอนแทคเซ็นเตอร์ อย่างมีคุณภาพ
- ค้นคว้าวิจัย พัฒนาเพื่อนำเทคโนโลยีมาประยุกต์ใช้ในธุรกิจของลูกค้า
- มีส่วนร่วมในการช่วยเหลือชุมชนและสังคม
- พัฒนากระบวนการ การทำงานที่มีคุณภาพอย่างต่อเนื่องทั่วทั้งองค์กร

- พัฒนาขีดความสามารถของบุคลากรอย่างต่อเนื่อง เพื่อมุ่งสู่องค์กรแห่งการเรียนรู้
- ส่งเสริมให้บุคลากรมีคุณภาพชีวิตในการทำงานที่ดี และมีความผูกพันต่อองค์กร

1.3 รูปแบบการจัดองค์กรและการบริหารองค์กร

1.3.1 แผนผังองค์กร



ภาพที่ 1.2แสดงแผนผังองค์กร

1.4 ตำแหน่งและหน้าที่งานที่นักศึกษาได้รับมอบหมาย

ตำแหน่ง: โปรแกรมเมอร์

หน้าที่งานที่ได้รับมอบหมาย: พัฒนาซอฟต์แวร์ตัวอย่าง โดยเขียนโปรแกรมด้วยภาษา Objective-C, Query บนโปรแกรม Xcode และใช้ฐานข้อมูล SQLite

1.5 พนักงานที่ปรึกษา และตำแหน่งของพนักงานที่ปรึกษา

ชื่อ :คุณอรวรา โรจนวาศ

ตำแหน่ง : Business Development

ชื่อ :นางอรุณวรรณ นาจารย์

ตำแหน่ง : Leader Business Development

ชื่อ : นายเฉลิมชัย สถานุรักษ์วงศ์

ตำแหน่ง : Software Engineering

1.6 ระยะเวลาที่ปฏิบัติงาน

เริ่มต้นการปฏิบัติงานสหกิจศึกษาวันที่ 3 มิถุนายน 2554 และสิ้นสุดการปฏิบัติงานสหกิจศึกษาวันที่ 30 กันยายน 2554

1.7 วัตถุประสงค์ของการปฏิบัติงาน

1) เพื่อให้การเก็บข้อมูลต่างๆ รวดเร็วถูกต้องเป็นมาตราฐานเดียวกัน
2) เพื่อให้การสั่งสินค้าเป็นไปได้อย่างรวดเร็ว
3) เพื่อลดการใช้ทรัพยากรกระดาษ

1.8 ผลที่คาดว่าจะได้รับจากการปฏิบัติงานหรือโครงงานที่ได้รับมอบหมาย

1) เพื่อเป็นแนวทางในการบริหารการจัดการสั่งสินค้าที่เป็นระบบ
2) เพื่อให้ดูเป็นองค์กรที่มีความน่าเชื่อถือ
3) เพื่อเป็นส่วนหนึ่งในการลดผลกระทบโลกร้อนในปัจุบัน

บทที่ 2
ทฤษฎีและเทคโนโลยีที่ใช้ในการปฏิบัติงาน

2.1 Software ที่ใช้ในการพัฒนา Application

-OS X (Snow Leopard, Lion), iOS(4.3), Window 7

-โปรแกรม Xcode (Version 4.0, 4.1)

-Web browserFirefox SQLite Manager Add-on

-โปรแกรม Visual Studio 2010

2.2Hardware ที่ใช้ในการพัฒนา Application

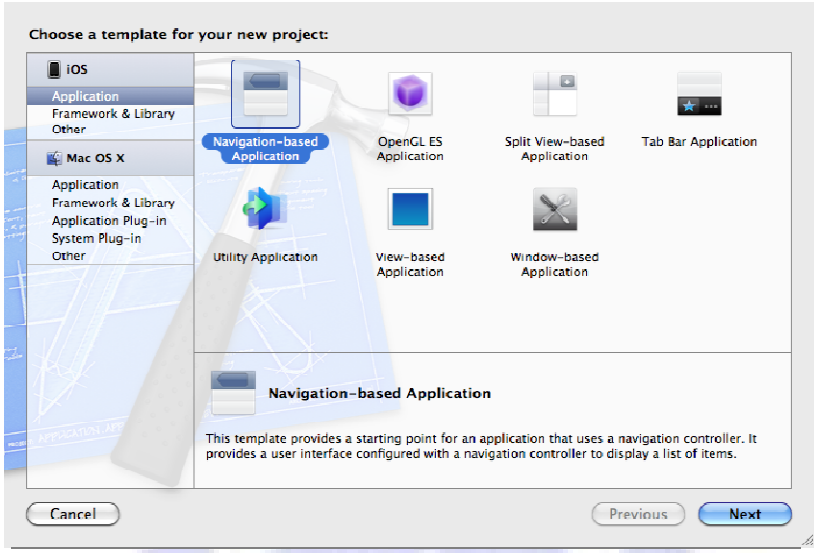
-iMac, iPhone, iPad

-Notebook

2.3 การใช้โปรแกรม xcode ในการเขียนโปรแกรม

Getting Start

เปิดโปรแกรม Xcode เลือก File > New > New Project จะมีหน้าต่างดังภาพที่ 2.1 ปรากฏขึ้น



ภาพที่ 2.1 Create new project

โดยมีประเภทต่าง ๆ ดังนี้

Navigation-based Application

คือการเริ่มโปรเจคโดย xcode จะสร้าง Navigation Controller และ table ให้

OpenGL ES Application

คือการเริ่มโปรเจค โดย xcode จะเพิ่ม Framework ที่จำเป็นต่อการใช้ OpenGLให้

Split View-based Application

คือการเริ่มโปรเจค โดย xcode จะสร้าง View ให้ 2 view โดย View ทางซ้ายเป็น Master pane ซึ่งเป็น Table View และ ทางขวาเป็น Detail pane ซึ่งเป็น View

Tab Bar Application

คือการเริ่มโปรเจค โดย xcode จะสร้าง Tab Bar Controller ให้

Utility Application

คือการเริ่มโปรเจค โดย xcode จะสร้าง View ที่มี info Button ที่สามารถเปลี่ยนหน้า ใน Present Modal View ได้

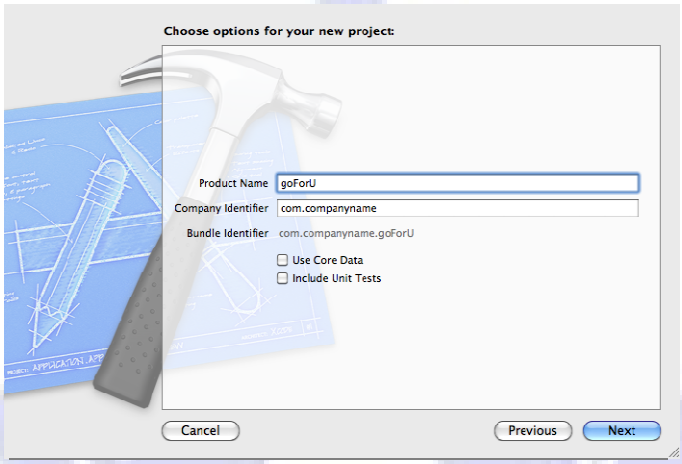
View-based Application

คือการเริ่มโปรเจคโดย xcode จะสร้าง View ที่มี View Controller ให้

Window-based Application

คือการเริ่มโปรเจค โดย xcode จะสร้างเฉพาะ AppDelegate ขึ้นมาให้เท่านั้น

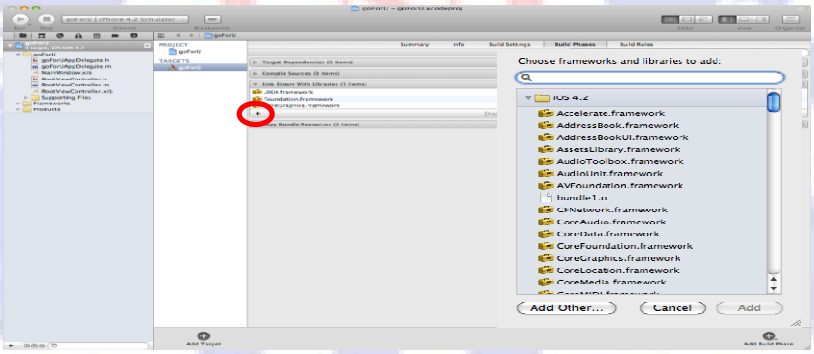
จากนั้นกด Next>ตั้งชื่อใน Product Name >กด Next >เลือก Location >กด Create



ภาพที่ 2.2 Options for new project

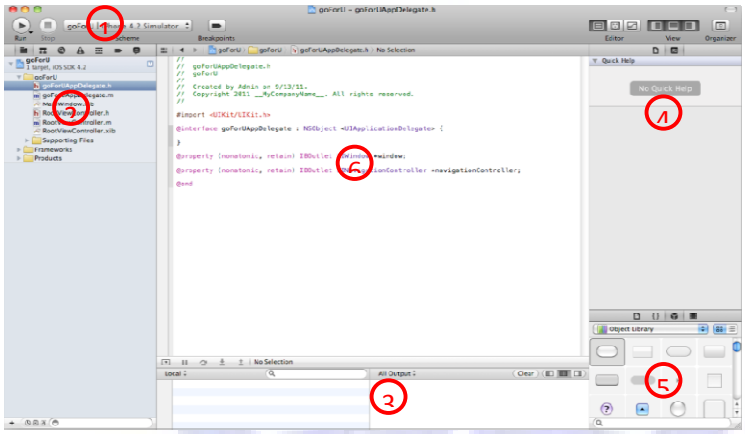
2.3.1 การเพิ่ม Framework

หากต้องการเพิ่ม Framework ที่เกี่ยวข้องเข้าไป จะต้องกดที่ ชื่อโปรเจคใน pane ด้านซ้าย >ชื่อโปรเจคใต้ Target> Build Phases > Link Binary With Libraries >เครื่องหมาย +



ภาพที่ 2.3 การเพิ่ม Framework เข้าไปใน Project

2.3.2 Xcode workspace



ภาพที่ 2.4 พื้นที่การทำงานของโปรแกรม xcode

1. **Toolbar** Run คือ Run Program คลิ๊กค้างไว้ จะมีRun, Test, Profile, Analyze

   Stop คือ ปุ่มหยุดการรัน

   Scheme คือ Target ของการรัน เช่น Simulator, iOS Device

   Breakpoint คือ การยกเลิก Breakpoint ทุกจุดในหน้านั้น ๆ

   Editor คือ การเลือก Page Control สำหรับ Workspace

   View คือ การแสดง/ซ่อน view อื่น ๆ

   Organizer คือ การจัดการ Apple Certification Authorization ของ Deviceที่เป็น Target
2. **Navigator** คือส่วนที่แสดงไฟล์และการDebugทั้งหมดในโปรเจค เช่น Header, Implementation, Framework, Function, Class, Method, Warning, Error, Breakpoint, etc.
3. **Debug Area** คือส่วนที่ใช้สำหรับ Debug แสดงว่าตัวแปรไหนเก็บค่าเท่าไหร่ไว้ หรือถ้าในโปรแกรมเขียนสั่งแสดง Log (NSLog) ไว้ ก็จะนำมาแสดงในส่วนนี้
4. **Utilities** คือส่วนที่แสดง Quick Help, Property ต่าง ๆ
5. **Libraries** คือส่วนที่แสดง Libraries ต่าง ๆ เช่น Object, Snippet, Media, Template Libraries

6. **Coding Area** คือส่วนที่เขียนโค้ดหลักการทำงานของโปรแกรม

2.3.3 การใช้ NeXT Interface Builder

การใช้ IB (Interface Builder) นั้น คือการใช้ Function ที่เป็นลักษณะ drag&drop ของ interface ของ xcode ตัวอย่างการใช้งานมีดังนี้

ประกาศตัวแปร

```
IBOutletUITableView *TableView;
```

หรือ

```
@property (nonatomic, retain) IBOutlet UITableView *TableView;
```

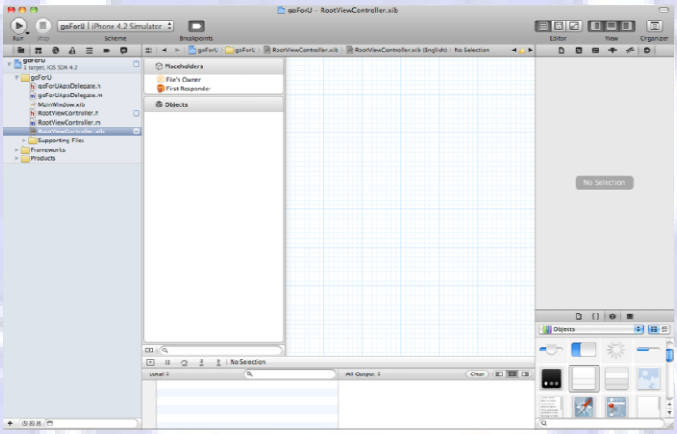
หากทำวิธีหลังต้องประกาศ @synthesize ในหน้า Implementation ด้วย

```
@synthesize TableView;
```

ประกาศ IBOutlet เพื่อให้ IB รู้จักตัวแปร

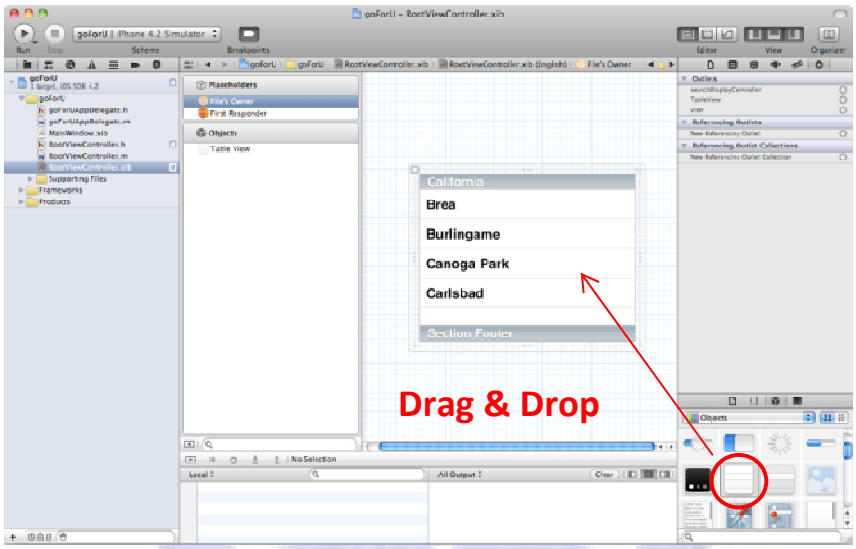
UITableView คือชนิดของตัวแปร

*TableView คือชื่อของตัวแปร



ภาพที่ 2.5 หน้า Interface Builder

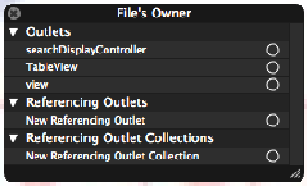
จากนั้นลาก TableView มาวางบนพื้นที่ว่าง



ภาพที่ 2.6 การแทรกตาราง

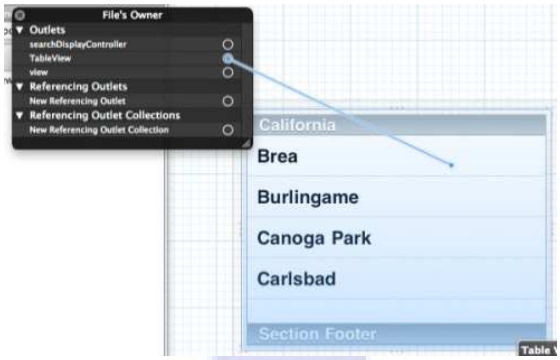
จากนั้นต้องกำหนด Property ให้ตารางดังนี้

1. คลิ๊กขวาที่ File's Owner จะปรากฏหน้าต่างดังภาพที่ "2.7"



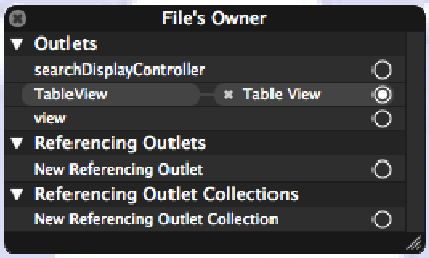
ภาพที่ 2.7 File's Owner Property

2.คลิ๊กลาก TableView ไปที่ Table ที่ลากเข้ามา



ภาพที่ 2.8 การกำหนด Property ให้ Object

3.จะปรากฏหน้าต่างดังภาพที่ 2.9

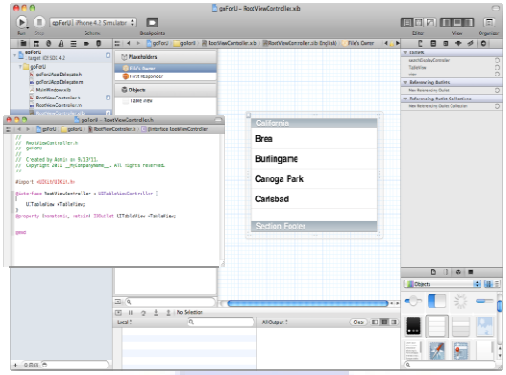


ภาพที่ 2.9 File Owner Property

2.3.4การประกาศแบบ วิธี Drag & Drop

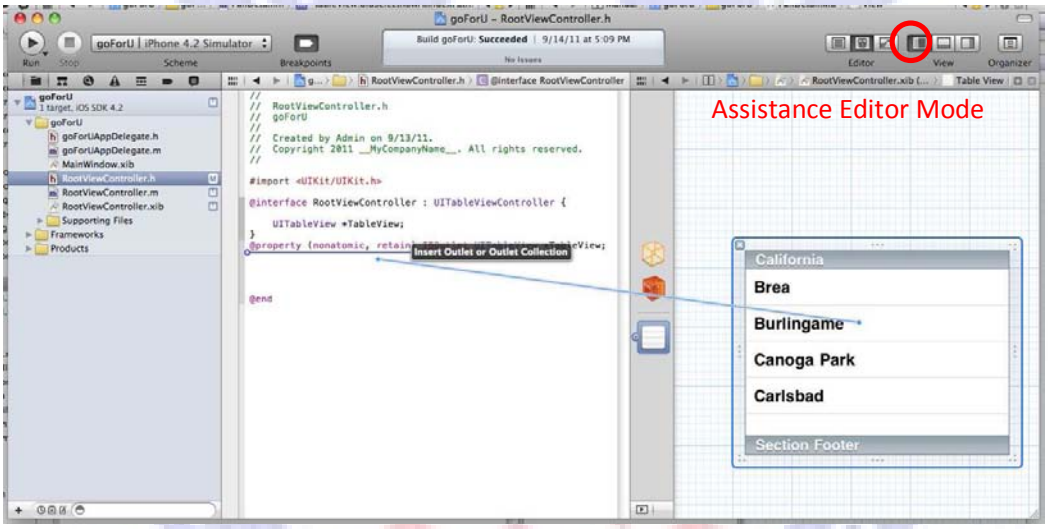
1.ลาก TableView มาวางบนพื้นที่ว่างจะปรากฏตารางดังภาพที่ 2.10

2.ดับเบิ้ลคลิ๊กที่ .h ของ File จะประกฏหน้าต่างดังภาพที่ 2.10



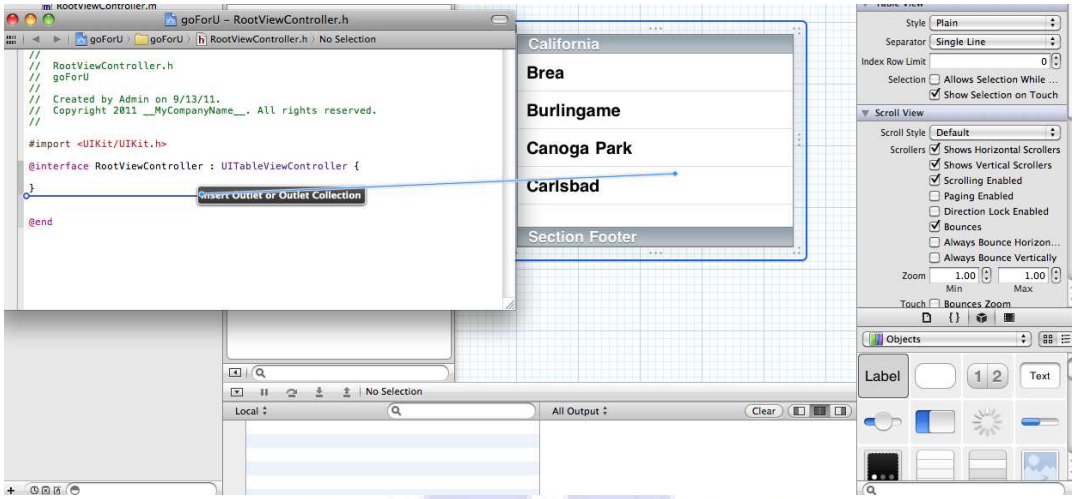
ภาพที่ 2.10 การเรียกหน้าต่างแยก

หรือเรียกใช้ Assistance Editor ภาพที่ 2.11



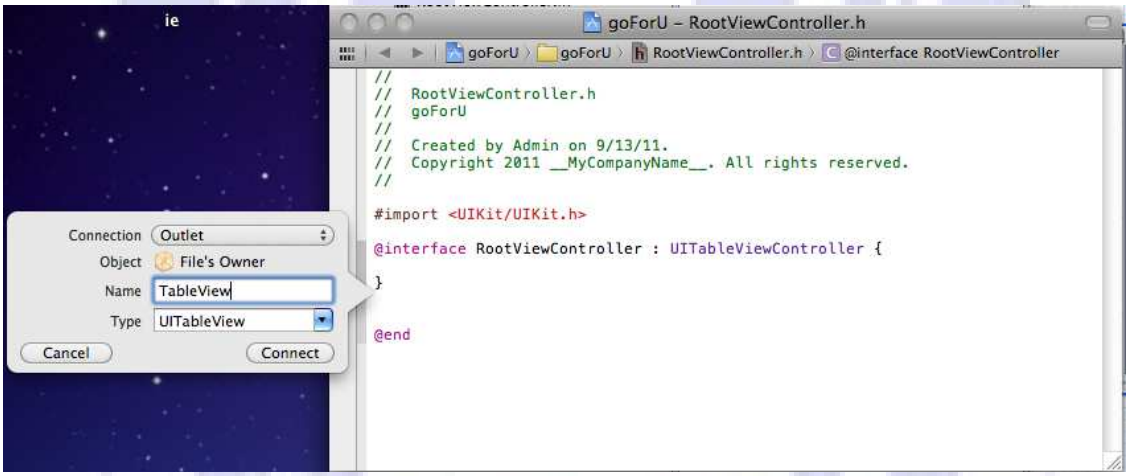
ภาพที่ 2.11 การลากโดยใช้ Assistance Editor

3.คลิ๊กขวาที่ Table ที่ แล้ว ลากไปที่ หน้าต่าง.h ที่เปิดขึ้นมาดังภาพที่ 2.12



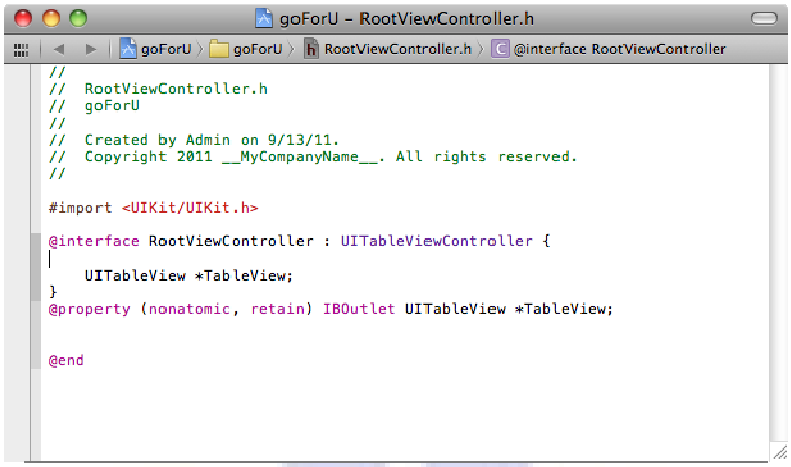
ภาพที่ 2.12 การลาก Object ไปที่ไฟล์ เพื่อกำหนด Property

4.จะปรากฏหน้าต่างดังภาพที่ 2.12 จากนั้นให้ตั้งชื่อ ตัวแปรที่ใช้ อ้างอิงถึง Object

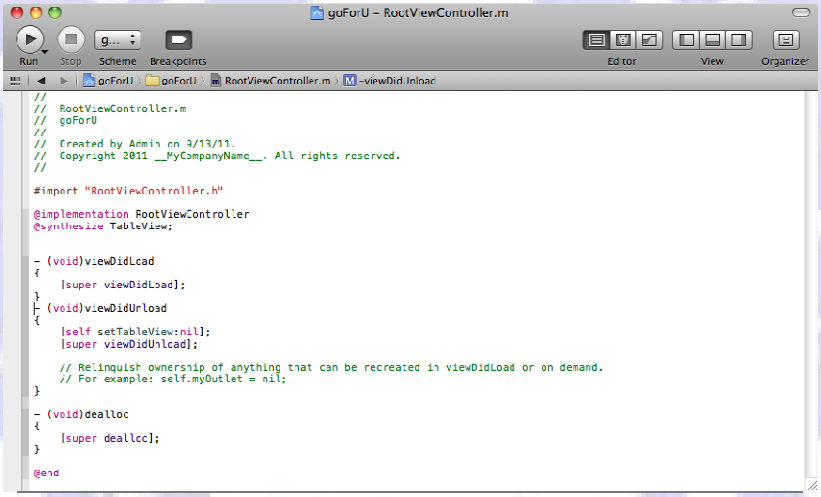


ภาพที่ 2.13 การตั้งชื่อตัวแปรของ Object

จากนั้น Program จะ ทำการสร้าง @property, ตัวแปร Local, @synthesize, release, delegate ให้โดยอัตโนมัติดังภาพที่ 2.13



ภาพที่ 2.14 การประกาศ Property แบบ drag & drop ของ Object หน้า .h



ภาพที่ 2.15 การประกาศ Property แบบ drag & drop ของ Object หน้า .m

2.4 พื้นฐานการใช้งานภาษา Objective-C ภายใน X-Code เบื้องต้น

2.4.1 การนำเพิ่ม Framework เข้ามาในไฟล์

```
#import <FrameworkName/FrameworkName.h>
```

โดยใน Framework ก็จะมี Class ที่สร้างมาให้สำเร็จสามารถนำไปใช้งานได้ทันที แต่จำเป็นที่จะต้องรู้ว่า Framework ไหนมี Class อะไรให้ใช้งานได้บ้าง ในส่วนนี้อาจจะต้องค้นหา จาก API Reference ที่มีอยู่ภายใน ProgramXcode ยกตัวอย่างเช่น<MapKit/MapKit.h>จะสามารถใช้งาน ตัวแปรที่เกี่ยวกับการทำโปรแกรมประยุกค์(Application)ที่ใช้ UIMapView ตัวอย่างเช่น ตัวแปรประเภท MKMapViewที่จะ สร้าง MapView ขึ้นมาโดยไม่จำเป็นต้องใช้ Interface Builder ของตัวโปรแกรม

2.4.2 การสร้าง Class และ การประกาศตัวแปร,ฟังก์ชันของ Class

ClassName.h (header)

```
@interface ClassName : ClassType<ProtocolReferenceName> {
   [1] VariableType *VariableName;
}

@property (nonatomic, retain) InterfaceBuilderTypeVariableType *VariableName;

-(void)FunctionName:(VariableType *)Parameter1 and:(VariableType *)Parameter2;
+(void)FunctionName:(VariableType *)Parameter1 and:(VariableType *)Parameter2;

@end
```

ClassName.m (implementation)

```
@implementation ClassName

-(void)FunctionName:(VariableType *)Parameter1 and:(VariableType *)Parameter2{
//function Method
};

+(void)FunctionName:(VariableType *)Parameter1 and:(VariableType *)Parameter2{
//function Method
};

@end
```

การสร้าง Class สามารถกระทำได้โดยประกาศ ชื่อคลาส ชนิดของคลาส (หากต้องการให้เป็น Class ที่ไม่ใช่ชนิดของคลาสใด ๆ ให้เราใส่ชนิดว่าเป็น NSObject) และ เราสามารถเพิ่ม Protocol ที่ต้องการ ใช้งานได้ตรงหลังชื่อชนิดของคลาสภายในสัญลักษณ์ <>เพิ่มให้ Class ที่สร้างขึ้น มีการตอบสนองต่อ Protocol เหล่านั้นเช่นเมื่อต้องการใช้งานUITableView ภายในคลาสที่ไม่ใช่ ประเภท UITableViewController จึงจำเป็นที่จะต้องกำหนด Protocol ให้กับ Class ตาม คำสั่งเช่น

```
@interface classSample : NSObject<UITableViewDelegate, UITableViewDataSource>
```

การประกาศตัวแปรที่ตำแหน่ง [1] คล้ายกับการประกาศ private ของ class ภายในภาษา C จะใช้งานได้เพียงแค่ภายใน implement ของ class ตัวเองเท่านั้นถ้าจะประกาศเป็น public จะสามารถ ประกาศตัวแปรเป็น property ดังในตัวอย่างข้างบนและต้องทำการ synthesize ในไฟล์ implementation

การสร้าง function ของ class จะมีการสร้างเป็นลักษณะเดียวกันกับตัวแปร ที่มีทั้ง private และ public โดยจะถูกบอกที่ก่อนหน้าชื่อฟังก์ชัน ชนิด private จะเป็น (-) และ public จะเป็น (+) จากนั้นจะสามารถสร้างตัวแปร parameter เพื่อรับค่าจากนอก function ได้เช่นเดียวกับภาษาซีดังเช่นตัวอย่างข้างต้นที่แสดงอยู่

หมายเหตุ: การประกาศตัวแปรภายใน Objective-C นั่นส่วนใหญ่ จะใช้เป็น Pointer ยกเว้นการประกาศตัวแปรชนิดที่ไม่ใช่ NSxxx หรือ UIxxx

2.4.3 การเรียกใช้งานFunction method

ตัวอย่าง

```
[selfmyFunction];

[selfviewDidload];

[NSStringstringWithFormat:@"%@", StringVariable];
```

สามารถเรียกใช้งานฟังก์ชันได้ดังตัวอย่างข้างต้นซึ่งในแต่ละ class ที่นำมาใช้งานมักจะมี Function ที่ทำมาให้อยู่แล้วเพื่อความสะดวกในการใช้งาน โดยถ้าต้องการเรียก Function ภายใน Class ตัวเอง จะสามารถทำได้โดยใช้คำว่าself เพื่อบอกว่าเป็นฟังก์ชันของมันเอง

2.4.4 การนำไฟล์ที่อยู่ภายใน Directory เดียวกันใน Project เพิ่มเข้ามาในไฟล์ปัจจุบัน

```
#import "ImplementName.m" or "HeaderName.h"
```

โดยสามารถทำการ import ได้ทั้งในไฟล์ Header และไฟล์ Implement

2.4.5 การประกาศชื่อ Class

```
@classClassName;
```

2.4.6 การสร้าง Class เป็นชนิดที่สร้างขึ้นเองภายใน Project

2.4.6.1สร้างโดย import ไฟล์จาก header อื่นมา

```
#import "OtherClassName.h"

@interface ClassName : ClassType<ProtocolReferenceName> {
OtherClassName *VariableName;
}
```

2.4.6.2 สร้างโดยประกาศชื่อ class

```
@classOtherClassName;

@interface ClassName : ClassType<ProtocolReferenceName> {
OtherClassName *VariableName;
}
```

ภายในไฟล์ ClassName.m

```
#import "OtherClassName.h"
@implementation ClassName

@end
```

การประกาศแบบนี้มีข้อแตกต่างกันตรงที่ถ้าไม่ทำการ import คลาสเข้ามาจะทำให้ตัวแปร ชนิด OtherClassName ไม่รู้จักโครงสร้างภายในของ class ประเภท OtherClassName จะรู้แต่เพียงว่า Other ClassName คือชื่อ class หนึ่งเท่านั้น

หากจำเป็นต้องใช้วิธีประกาศชื่อ class แต่อยากให้มัน รู้ว่าโครงสร้างภายใน class เป็นอย่างไร สามารถทำได้โดยการ import ไฟล์ .h ของ class นั้นไปยังไฟล์ .m ที่เราต้องการใช้งาน ดังเช่นตัวอย่างข้างต้น(โดยปกติที่นิยมกันจะประกาศตัวแปรด้วยวิธีสร้างโดยประกาศชื่อคลาส)

2.4.7 การจองพื้นที่ใน memory และการคืนค่า memory พื้นฐาน

การจอง memory

```
VariableType *VariableName = [[VariableTypealloc] init];
```

การคืน memory

```
[VariableName release];
```

ในการจองพื้นที่ให้กับตัวแปรเราจำเป็นต้องจอง memory กับตัวแปรที่เป็น pointer และเมื่อ มีการใช้งานเสร็จไม่ควรที่จะลืมคืนค่านั้นทิ้งไป เพราะอาจทำให้เป็นสาเหตุของการเกิด memory leak ได้

2.4.8 การกำหนดค่า property ต่าง ๆ ให้กับตัวแปรที่เราสร้างขึ้น

```
VaribleName = 13.727775;
ClassName.PropertyName = NO;
VariableName = [UIColorcolorWithWhite:Valuealpha:Value];
[UIViewVariableName setFrame:CGRectMake(yPoint, xPoint, ySize, xSize)];
```

การกำหนดค่าให้กับตัวแปรต่าง ๆ สามารถทำได้ในหลาย ๆ วิธีในภาษา Objective-C เช่นเดียวกับภาษา C คือสามารถใช้ Expression (=) ในการแอดค่าได้ทันที หรือจะใช้ Function ที่สร้างขึ้นมาใน ตัวแปรชนิดนั้น ๆ กำหนดค่าก็ได้ดังตัวอย่างที่แสดงข้างต้น

2.4.9หลักการใช้งาน Delegate เพื่อเข้าถึง interface ทั้งหมดภายใน Project

ในการเขียน Application บางครั้งเราต้องการที่จะเรียกถึงฟังก์ชันในหน้าอื่น ๆ แต่ไม่สามารถเรียกได้ ซึ่งอาจมีเหตุผลมาจาก Class ที่มีฟังก์ชันที่ต้องการเรียกเป็น Class แม่ของ Class ที่ทำงานอยู่สามารถประกาศ AppDelegate ขึ้นมาเพื่อใช้ในการเข้าถึงโดยมีวิธีการดังนี้

วิธีการใช้งาน Delegate

1. ทำการ import ไฟล์ AppDelegate.h ของ Project เข้ามา
2. ประกาศตัวแปรขึ้นมาเป็นประเภท AppDelegate เพื่อใช้ในการ Share Delegate ตาม syntax นี้

```
AppDelegate *delegate = [[UIApplicationsharedApplication]delegate];
```

3. เรียก subclass ผ่าน delegate ไปจนถึงไฟล์ที่ต้องการ (จำเป็นต้อง import ไฟล์ class ที่จะเข้าถึง มาด้วยเพื่อให้ class ปัจจุบันรู้จักว่ามีตัวแปรนั้น ๆ อยู่ใน class ที่จะเข้าถึง)

ตัวอย่างการใช้งาน

TestAppDelegate.h

```
#import <UIKit/UIKit.h>
@classTestViewController;
@interface TestAppDelegate : NSObject<UIApplicationDelegate> {
}
@property (nonatomic, retain) IBOutletUIWindow *window;
@property (nonatomic, retain) IBOutletTestViewController *viewController;
@end
```

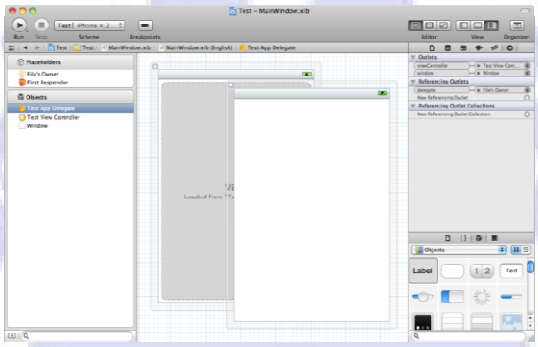
หมายเหตุ :IBOutlet คือการประกาศเพื่อให้ทาง Interface Builder รู้ว่ามี Outlet ชื่อนี้อยู่เพื่อนำไป โยงเข้ากับ UI นั้น ๆ

TestAppDelegate.m

```
#import "TestAppDelegate.h"
#import "TestViewController.h"
@implementation TestAppDelegate
@synthesize window=_window;
@synthesize viewController=_viewController;
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary
*)launchOptions{
self.window.rootViewController = self.viewController;
    [self.windowmakeKeyAndVisible];
returnYES;
```

```
}
- (void)dealloc{
    [_windowrelease];
    [_viewControllerrelease];
    [superdealloc];
}
@end
```

TestAppDelegate.xib



ภาพที่ 2.16แสดงหน้าจอ NIB ของ TestAppDelegate

RootViewController.h

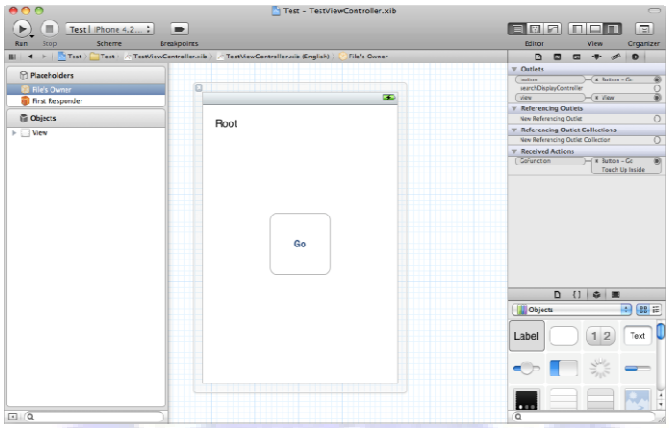
```
#import <UIKit/UIKit.h>
@classOtherViewController;
@interface TestViewController : UIViewController
@property (nonatomic,retain) OtherViewController *otherViewController;
@property (nonatomic,retain) IBOutletUIButton *button;
- (IBAction)GoFunction;
- (void)CallFunction;
@end
```

RootViewController.m

```
#import "TestViewController.h"
#import "OtherViewController.h"
@implementation TestViewController
@synthesize otherViewController;
@synthesize button;
- (void)dealloc{
    [superdealloc];
    [otherViewControllerrelease];
    [buttonrelease];
}
```

```
- (IBAction)GoFunction {
otherViewController = [[OtherViewControlleralloc]initWithNibName:@"OtherViewController"
bundle:nil];
    [selfpresentModalViewController:otherViewControlleranimated:YES];
}
- (void)CallFunction {
otherViewController.label.text = @"Hello World!";
}
@end
```

RootViewController.xib



ภาพที่ 2.17แสดงหน้าจอ NIB ของ RootViewController

OtherViewController.h

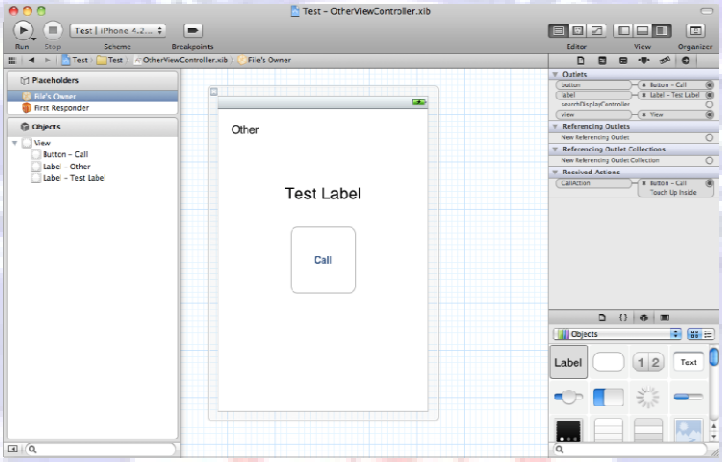
```
#import <UIKit/UIKit.h>
@interface OtherViewController : UIViewController
@property (nonatomic,retain) IBOutletUIButton *button;
@property (nonatomic,retain) IBOutletUILabel *label;
-(IBAction)CallAction;
@end
```

OtherViewController.m

```
#import "OtherViewController.h"
#import "TestAppDelegate.h"
#import "TestViewController.h"
@implementation OtherViewController
@synthesize button,label;
- (void)dealloc{
```

```
    [superdealloc];
    [buttonrelease];
}
-(IBAction)CallAction {
TestAppDelegate *appDelegate = [[UIApplicationsharedApplication]delegate];
    [appDelegate.viewControllerCallFunction];
}
@end
```

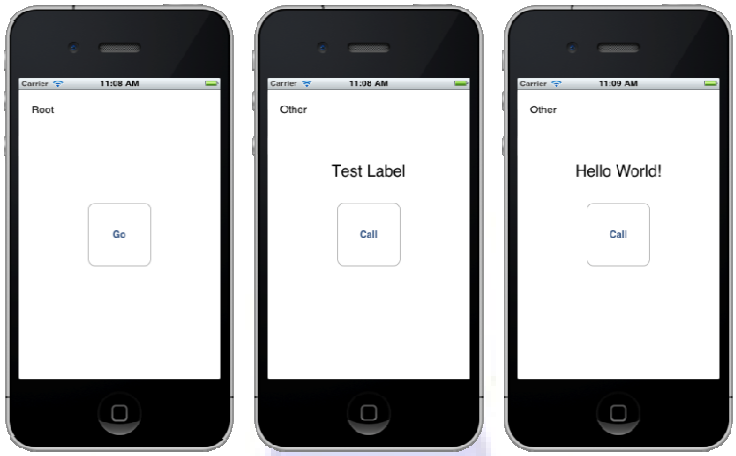
OtherViewController.xib



ภาพที่ 2.18แสดงหน้า NIB ของ OtherViewController

ขั้นตอนการทำงานของ App. ตัวอย่าง

(เริ่มที่กดปุ่มหน้า Root >กดปุ่มที่หน้า Other> เรียกฟังก์ชันใน Root)



ภาพที่ 2.19แสดงinterface ในแต่ละ Step การทำงาน

2.4.10 UI Class (User Interface Class) พื้นฐานภายในโปรแกรม Xcode

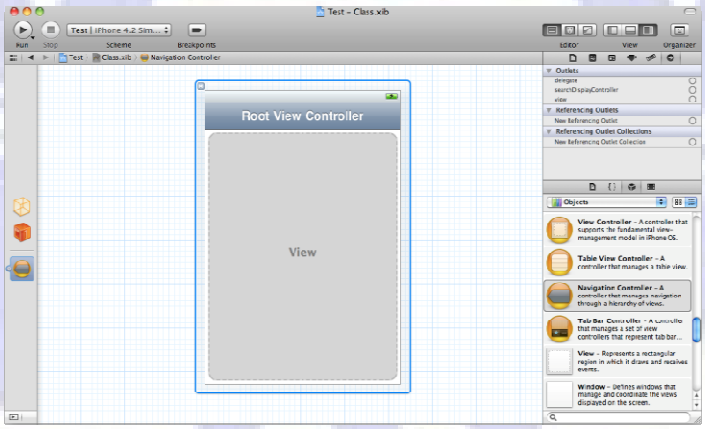
ภาพที่ 2.20แสดงหน้าข้อมูล Interface Builder

ในการเขียน Application นั้นการสร้าง Interface ของ Application แบ่งออก เป็น 2 วิธี คือ การเขียนด้วยโค้ด (programmatically) หรือการใช้อุปกรณ์สำเร็จรูปที่สามารถใส่ Interface ได้เลย (NeXT Interface Builder หรือไฟล์ .xib) โดยการสร้างทั้ง 2 แบบจะให้ผลลัพธ์ที่เหมือนกันกับ การใช้โค้ดเขียน Interface ขึ้นมา จะมีความยุ่งยากมากกว่าการใช้ NIB (NeXT Interface Builder) แต่ จะมีข้อดีตรงที่มีความยืดหยุ่นมากกว่า สามารถปรับแต่งในรูปแบบที่มีความเฉพาะตัวได้

การสร้าง Interface นั้นจะมี Controller ที่เปรียบเสมือนการวางแผ่นรองลงไปบนพื้นก่อนที่จะเริ่มทำการวาง Interface อื่น ๆ ลงไปยังหน้า Application โดยหลัก ๆ ที่ใช้ก็จะมีUIViewController, UITableViewController, UITabBarController, UISearchBar และ UINavigationController หากต้องการเพิ่มการทำงานของ Controller นั้นเราสามารถเพิ่ม Protocol อื่น ๆ ลงไปได้อีกด้วย เช่นเมื่อเพิ่ม <UIAlertViewDelegate>จะทำให้ Controller นั้นสามารถใช้งาน AlertView Protocol ได้ ลำดับของการใส่ Controller เป็น Sub ลำดับได้ดังนี้ TabBarController > NavigationController > TableViewController = ViewController และใน TableViewController> SearchBarController

ในโปรแกรม Xcode จะมี Interface ต่าง ๆ ให้เราใช้งานซึ่งหากต้องการใช้ Interface ใด ๆ ก็สามารถเข้าไปที่ API Reference เพื่อศึกษาวิธีการใช้งานได้ด้วยตนเอง โดยเราจะยกตัวอย่าง Interface ที่เป็น Controller พื้นฐานที่ใช้งานภายใน Project ที่มีหน้าที่สำคัญ ๆ ดังนี้

2.4.10.1 UINavigationController & UINavigationBar



ภาพที่ 2.21 แสดงUINavigationController ในหน้า NIB

การทำงานของ UINavigationController เป็น Controller ที่มีหน้าที่หลักในการพาผู้ใช้งาน ไปยังหน้า View อื่น ๆ และสามารถทำการกลับไปเป็นลำดับเพื่อกลับไปยังหน้า View แรกได้ โดยตัว NavigationBar นั้นสามารถที่จะกำหนดค่าข้อความของ Title เพื่อให้ทราบว่าอยู่ที่หน้าใดหรือเพื่อเป็นการบอกว่า Application นี้มีชื่อว่าอะไร รวมไปถึงการเซ็ทค่าของชนิด NavigationBar และอื่น ๆ

ตัว NavigationBarสามารถทำการใส่ NavigationBarItem ยกตัวอย่างเช่นการใส่ปุ่ม Setting,Cancel, Done เป็นต้นโดยปกติแล้วหากเราไม่ได้ทำการใส่ปุ่มลงใน NavigationBar ตัวมันเองจะมีปุ่ม Back อยู่ทางซ้ายหากหน้า View นั้นทำการ Push มาจากหน้าใด ๆ

การ Push/PopNavigation เพื่อไปยัง View หน้าต่อไปหรือกลับไปยัง View ก่อนหน้า

```
[navController pushViewController:otherViewController animated:YES];
[otherViewController.navigationControllerpopViewControllerAnimated:YES];
```

ตัวอย่างการ Setting ค่าเบื้องต้นของ UINavigation

```
navController.title = @"TitleName";
navController.navigationBar.tintColor = [UIColorblackColor];
```

การสร้าง UINavigationController

```
UIViewController *rootViewController = [[UIViewControlleralloc]init];
UINavigationController *navController = [[UINavigationControlleralloc]
initWithRootViewController:rootViewController];
```

หมายเหตุ : การสร้าง NavigationController ปกติเราจะใส่ ViewController หรือ TableViewController อย่างใดอย่างหนึ่งลงเป็น RootViewController เพื่อแสดงผลในรูปแบบที่ต้องการ

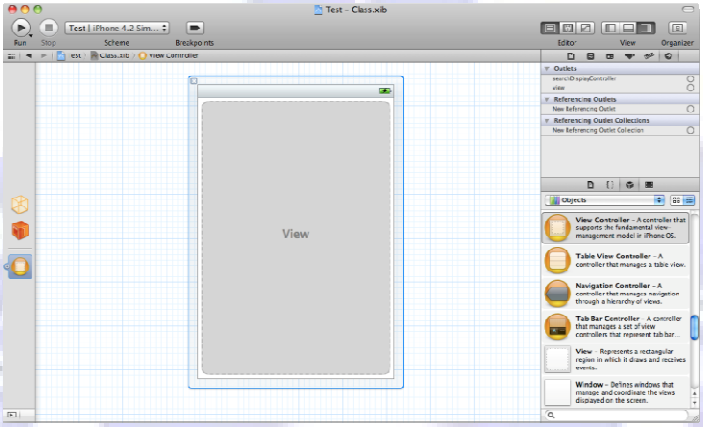
การสร้างปุ่มบน UINavigationBar

```
UIBarButtonItem *rightButton = [[UIBarButtonItemalloc]
initWithTitle:@"Edit"
style:UIBarButtonSystemItemEdit
```

```
target:self
action:@selector(EditFunction)];
navController.navigationItem.rightBarButtonItem = rightButton;
[rightButton release];
```

ในการสร้างปุ่มนั้นสามารถสร้างปุ่มบน NavigationBar โดยจะสามารถใส่ทางซ้ายและขวาข้างละ1ปุ่ม

2.4.10.2 UIViewController & UIView



ภาพที่ 2.22แสดงUIViewController ในหน้า NIB

เป็น UI ที่มีหน้าที่ในการควบคุมการแสดงผลของUIView โดยเราสามารถวาง interface ต่าง ๆ ลงไปบนUIViewController ได้ ซึ่งการสร้าง Controller โดยโค้ดจะสามารถทำได้โดย syntax ดังนี้

สร้าง ViewController ปกติ

```
UIViewController *viewController = [[UIViewControlleralloc]init];
```

สร้าง Class ประเภท ViewController

```
@interface myViewController : UIViewController
```

หากสร้าง Class ที่เป็นชนิด ViewController ภายในของ ViewController จะมี Method ที่จะทำงานในแต่ละ State ของการทำงานหลัก ๆ ดังนี้

ViewController View Life Cycle

```
- (void)viewDidLoad
- (void)viewWillAppear
- (void)viewDidAppear
- (void)viewWillDisappear
- (void)viewDidDisappear
- (void)viewDidUnload
etc.
```

โดยในแต่ละ Function จะถูกทำงานเป็นลำดับแตกต่างกัน ซึ่งในจุดนี้จะเป็นจุดที่ใส่ code ของการทำงานลงไปเช่นหากต้องการให้ code ใด ๆ เริ่มทำงานก่อนหน้าที่ ViewController นี้จะแสดงผล ก็จะใส่ code นั้นไปที่ viewWillAppearที่เป็น Function ที่ถูกเรียกทุกครั้งเมื่อจะมีการแสดงผลหน้า Controller ซึ่งต่างจาก ViewDidLoad ซึ่งจะรันเพียงครั้งเดียวหลังจากที่ Run Application และหากต้องการให้เมื่อ ViewController นั้นแสดงผลแล้วให้ทำการเริ่มการทำงาน code ก็ให้ใส่ลงใน FuntionviewDidAppear เป็นต้น

การจัดการคืนค่า memory

```
- (void)dealloc
```

การคืนค่า memory ของโปรแกรมสามารถกระทำได้โดยการเขียนโค้ดคืนค่า ภายใน Function dealloc เพื่อให้ เมื่อจบการทำงานของ application แล้ว ส่วนของ memory ที่ถูกจองค่าไว้จะมีการปล่อยคืนเพื่อป้องกันไม่ให้เกิดปัญหา Memory Leak

การนำ View ไปวางบน ViewController.view

```
[1] [viewController.viewaddSubview:MyView];
[2] [viewController.viewinsertSubview:MyView aboveSubview:MyViewBG];
[3] [viewController.viewinsertSubview:MyView belowSubview:MyViewBG];
```

เมื่อต้องการใส่ Object อื่น ๆ ลงไปบนUIViewController ที่สร้างขึ้นสามารถใส่คำสั่ง เพื่อให้ Object เหล่านั้นไปวางอยู่บน Controller ได้ด้วยคำสั่งข้างต้นซึ่งแต่ละคำสั่งมีการทำงานดังนี้

[1] คือการ add MyView ลงไปยัง view โดยจะทับลงไปบน view เรื่อย ๆ หากมีการ add ซ้ำอีกครั้ง

[2] คือการ addMyView ลงไปยัง view แต่จะอยู่เหนือ view ที่เรากำหนด (ในที่นี้คือ MyViewBG)

[3] คือการ add MyView ลงไปยัง view แต่จะอยู่หลัง view ที่เรากำหนด (ในที่นี้คือ MyViewBG)

การกำหนดขนาดและตำแหน่ง UIView

```
[MyView setFrame:CGRectMake(float x, float y, float width, float height)];
MyView.frame = CGRectMake(float x, float y, float width, float height);
```

การกำหนดขนาด, ตำแหน่งของ UIView ได้ เพื่อให้ View ของเรามีขนาด, ตำแหน่ง ตามที่ต้องการได้โดยจะมีความสำคัญมากในเวลาที่เราต้องการสร้างหน้า Interface ที่มีความเฉพาะตัว ซึ่งอาจไม่สามารถกำหนดได้ในหน้า NIB

การกำหนด ModalPresentationStyle

```
[1]rootViewController.modalTransitionStyle = UIModalTransitionStyleCoverVertical;
[2]rootViewController.modalTransitionStyle = UIModalTransitionStyleCrossDissolve;
[3]rootViewController.modalTransitionStyle = UIModalTransitionStyleFlipHorizontal;
[4]rootViewController.modalTransitionStyle = UIModalTransitionStylePartialCurl;
```

โดยแต่ละคำสั่งมีลักษณะการทำงานดังนี้

[1] จะแสดงหน้าที่เรียกด้วยการแทรกจากด้านล่างของจอแสดงผล

[2] จะแสดงหน้าที่เรียกด้วยการค่อย ๆ เข้มขึ้นมาทับหน้าปัจจุบัน (Fade)

[3] จะแสดงหน้าที่เรียกด้วยการพลิกหน้าปัจจุบันมาเป็นหน้าที่เรียก

[4] จะแสดงหน้าที่เรียกด้วยการพลิกในลักษณะการเปิดหนังสือ

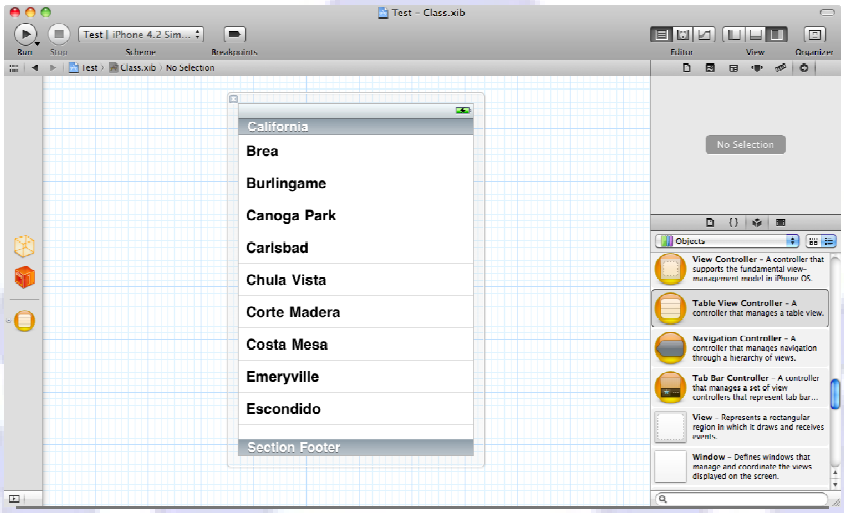
การ presentModalViewController และการ dismissModalViewController

```
[rootViewController presentModalViewController:otherViewController
animated:YES];
```

```
[otherViewController dismissModalViewControllerAnimated:YES];
```

การ presentModalViewController เปรียบเสมือนการแทรกหน้าของ ViewController อื่นขึ้นมาทับบน หน้า Controller ปัจจุบันด้วยการ AddSubView ส่วนการ dissmissModalViewController ก็เสมือนกับ การ removeFromSuperView นั่นเอง

2.4.10.3 UITableViewController & UITableView



ภาพที่ 2.23แสดงUITableViewController ในหน้า NIB

เป็น Controller ที่จะแสดง UITableView ออกมาเป็นหน้าหลักและมี Method ในการทำงานเช่นเดียวกับ ViewController ที่จะมี Method ในการเช็ค state จำพวก ViewDidLoad, ViewDidUnload,ViewWillAppear ที่เหมือนกันแต่จะมีเพิ่มขึ้นมาในส่วนของ UITableViewController โดย Method ที่ใช้งานหลัก ๆ มีดังนี้

จำนวน Section ของ TableView ภายใน TableViewController

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView{
```

```
return0;
}
```

หมายเหตุ : Section คือหัวข้อภายใน Table ดังภาพข้างต้นคือส่วนแถบที่ชื่อว่า Callifornia เราสามารถกำหนดค่าของชื่อหัวข้อและขนาดของหัวข้อได้โดยการใส่ function ลงไปเพิ่มเช่น

ตัวอย่าง - การกำหนดค่าความกว้างและชื่อของหัวข้อ (Section)

```
-(CGFloat)tableView:(UITableView *)tableView heightForHeaderInSection:(NSInteger)section{
return10.0;
}
-(NSString *)tableView:(UITableView *)tableView titleForHeaderInSection:(NSInteger)section{
return@"TableTitleText";
}
```

จำนวน Row ในแต่ละ Sectionของ TableView ภายใน TableViewController

```
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section{
return0;
}
```

ตั้งค่า Cell ของ TableView ภายใน TableViewController

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath
*)indexPath{
staticNSString *CellIdentifier = @"Cell";
UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
if (cell == nil) {
      cell = [[[UITableViewCellalloc]
initWithStyle:UITableViewCellStyleDefaultreuseIdentifier:CellIdentifier] autorelease];
   }
```

```
return cell;
}
```

เป็นการสร้าง Cell ภายใน UITableView โดยหลักการทำงานคือจะมีการเรียกใช้งานฟังก์ชั่นนี้ในทุก ๆ ครั้งที่มีการขยับหน้าจอจนทำให้มองเห็น Cell ที่ก่อนหน้านี้ไม่แสดงผลในหน้าจอเพื่อสร้าง Cell ขึ้นมา ให้ผู้ใช้งานมองเห็นได้นั่นเอง

การใช้งาน Custom Cell

```
#import <UIKit/UIKit.h>
@interface MyCell : UITableViewCell{
IBOutletUILabel *label1;
IBOutletUILabel *label2;
IBOutletUILabel *label3;
}
-(void)setLabel1:(NSString *)l1;
-(void)setLabel2:(NSString *)l2;
-(void)setLabel3:(NSString *)l3;
@end
```

ในหน้า Custom Cell.h ให้ประกาศ Object ที่ต้องการจะใช้ใน Cell โดยมี IBOutlet ด้วยเมื่อต้องการใช้ Interface Builder จากนั้นจึงประกาศ Function เพื่อ Set ค่าให้ Object

เมื่อประกาศ Object หน้า .h เสร็จแล้ว ต้อง link ตัวแปรที่ประกาศกับ Object ในหน้า xib ด้วย

ภาพที่ 2.24 แสดง Custom Cell

จากนั้นก็มาเขียนฟังก์ชั่น Set เพื่อใส่ค่าให้ Object ดังนี้

```
#import "MyCell.h"
@implementation MyCell
-(void)setLabel1:(NSString *)text1{
label1.text = text1;
}
-(void)setLabel2:(NSString *)text2{
label2.text = text2;
}
-(void)setLabel3:(NSString *)text3{
label3.text = text3;
}
@end
```

การตั้งค่าให้สามารถทำการ Edit Cell ได้

```
- (BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:(NSIndexPath
*)indexPath{
returnYES;
}
```

โดยปกติแล้วเราสามารถสไลด์นิ้วไปบน Cell เพื่อทำการ Edit (Delete) Cell โดยสามารถกำหนดค่าให้เป็น NO ในกรณีที่ไม่ต้องการให้มีการ Edit หรือ YES เพื่อให้สามารถทำการ Edit ได้

เมื่อมีการ Edit TableView ให้แสดงผลรูปแบบการ Edit เป็นตามที่ตั้งค่าไว้

```
- (void)tableView:(UITableView *)tableView
commitEditingStyle:(UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:(NSIndexPath
*)indexPath
{
if (editingStyle == UITableViewCellEditingStyleDelete) {
```

```
    [tableView deleteRowsAtIndexPaths:[NSArrayarrayWithObject:indexPath]
withRowAnimation:UITableViewRowAnimationFade];
  }
elseif (editingStyle == UITableViewCellEditingStyleInsert) {
  }
}
```

ในการ Edit แต่ละครั้งของ CellController จะรู้ว่ามีการ Edit แต่จะยังไม่ทราบว่าให้ทำอะไร ต่อไป ซึ่งเราสามารถกำหนดคำสั่งที่อยากให้ทำงานได้เองด้วยการนำ Code ที่เราต้องการใส่ลงมาภายใน Function นี้โดยกำหนดได้ว่าหากจะทำการ Delete ให้ทำอย่างไร หากทำการ Insert ให้ทำอย่างไรที่ cell ไหนเป็นต้น

เมื่อกดเลือก Cell ของ TableView ใน TableViewController

```
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
  [tableView deselectRowAtIndexPath:indexPath animated:YES];
}
```

Function นี้ใช้ในการตรวจจับการสัมผัสที่ TableViewCell ภายใน TableView ว่าเลือกที่ Cell ไหน ของ Table และ เราสามารถที่จะกำหนดให้มีการทำงานใดต่อไปภายใน Function นี้นั่นเอง เช่นเมื่อเลือก Row ที่ 1 ของ section 2 ให้ทำการ PushView ไปยังหน้าแสดงรายละเอียดสินค้า เป็นต้น

2.5 Objective-C

เป็นภาษาโปรแกรมเชิงวัตถุและมีสมบัติการสะท้อนโดยแรกเริ่ม ภาษาอ็อบเจกทีฟ-ซี พัฒนาขึ้นจากภาษาซีโดยยังคงคุณลักษณะของภาษาซีไว้ครบทุกประการเพียงแต่เพิ่มระบบส่งข้อความ (messaging) แบบเดียวกับภาษาสมอลล์ทอล์กเข้าไปเท่านั้น (Objective-C runtime) ปัจจุบันภาษาอ็อบเจกทีฟ-ซีมีคุณสมบัติอื่น ๆ เพิ่มเติมจากการพัฒนาภาษาอ็อบเจกทีฟ-ซี 2.0 โดยบริษัทแอปเปิล

ปัจจุบันภาษาอ็อบเจกทีฟ-ซีถูกใช้มากใน Cocoa (API) ใน Mac OS X, GNUstep (API) และ Cocotron (API) เป็นต้น ซึ่งระบบเหล่านี้ได้รับการพัฒนาขึ้นโดยมีพื้นฐานจากมาตรฐาน OpenStep (API) ใน Nextstep (Operating system) โดยมีภาษาภาษาอ็อบเจกทีฟ-ซีเป็นภาษาหลัก ปัจจุบัน Mac OS

Xใช้ Cocoa เป็นเฟรมเวิร์กสำหรับสร้างโปรแกรมประยุกต์ โดยไลบรารีและ/หรือ API เหล่านี้เป็นเพียงส่วนเพิ่มขยาย (Software extension) เท่านั้น โปรแกรมที่ใช้ภาษาอ็อบเจกทีฟ-ซีทั่วไปที่ไม่ได้ใช้ส่วนเพิ่มขยายเหล่านี้ก็ยังสามารถคอมไพล์ได้ เช่นอาจใช้แต่ gcc ซึ่งรองรับภาษาอ็อบเจกทีฟ-ซี

2.5.1Syntax

ภาษาอ็อบเจกทีฟ-ซีเป็นชั้นบาง ๆ บน C และเป็น*สตริกต์* superset ของ C ดังนั้นคอมไพเลอร์ภาษาอ็อบเจกทีฟ-ซีจึงสามารถคอมไพล์โปรแกรมภาษา C ใด ๆ ก็ได้ ภาษาอ็อบเจกทีฟ-ซีได้รับรูปแบบการเขียนมาจากภาษาซีและภาษาสมอลล์ทอล์ก โดยรูปแบบส่วนใหญ่ (preprocessing, expressions, การประกาศฟังค์ชัน และการเรียกฟังค์ชัน)มาจากภาษาซี ขณะที่ส่วนที่เป็นการจัดการเชิงวัตถุมาจากสมอลทอล์ค

2.5.1.1 Message

ภาษาอ็อบเจกทีฟ-ซีได้เพิ่มเติมรูปแบบการเขียนโปรแกรมเพื่อรองรับการออกแบบโปรแกรมเชิงวัตถุ โดยจะใช้การส่ง message ไปยัง object ต่าง ๆ เช่นเดียวกับสมอลทอล์ค ซึ่งแตกต่างจากภาษาในตระกูล Simula (เช่น C++) ข้อแตกต่างมีมีความสำคัญ เพราะภาษาอ็อบเจกทีฟ-ซีจะไม่เรียก method แต่จะส่ง message

ในภาษาอ็อบเจกทีฟ-ซี ถ้ามี object หนึ่งชื่อ obj โดย class มี method ชื่อ doSomething หมายความว่า obj *respond*หรือตอบสนองต่อ message doSomething และถ้าเราต้องการจะส่ง message doSomething ไปยัง obj เราจะเขียนคำสั่งดังนี้

```
[obj doSomething];
```

ขณะที่ถ้าเป็น C++ เราจะเขียนว่า

```
Obj.dosomething();
```

2.5.1.2 Forwarding

ในภาษาอ็อบเจกทีฟ-ซีจะยอมให้มีการส่ง message ไปยัง object ใด ๆ แม้ว่าจะไม่มีการเตรียม method เอาไว้รองรับ (คือไม่ respond) ต่างจากภาษาอื่น ๆ เช่น C++ หรือ Java ที่การเรียกใช้ method

ต้องมีการระบุไว้ล่วงหน้า หาก object ได้รับ message ที่ไม่รู้จัก object จะผ่านต่อ message ที่ได้รับไปยัง method เหล่านี้

-(retval_t) forward:(SEL) sel :(arglist_t) args; // with GCC

-(id) forward:(SEL) sel :(marg_list) args; // with NeXT/Apple systems

2.5.1.3 อินเตอร์เฟซ และ อิมพลีเมนเทชัน

ในภาษาอ็อบเจกทีฟ-ซี ส่วนอินเตอร์เฟซ (@interface) และอิมพลีเมนเทชัน (@implementation) จะถูกแยกออกจากกัน ในทางปฏิบัติเรามักเก็บส่วนอินเตอร์เฟซไว้ในแฟ้ม .h และส่วนอิมพลีเมนเทชันใน .m

@interface

เรามักนิยามส่วนอินเตอร์เฟซของคลาสในแฟ้ม .h โดยทั่วไปเรามักตั้งชื่อแฟ้มนี้ให้ตรงหรือสอดคล้องกับชื่อของคลาส เช่นถ้าคลาสเราชื่อThingเราก็มักจะประกาศอินเตอร์เฟซของคลาสThingในแฟ้มThing.h

รูปแบบของการประกาศอินเตอร์เฟซมีลักษณะดังนี้:

```
@interface classname : superclass name
{
instance variables
}
+ classMethod1;
+(return_type) classMethod2;
+(return_type) classMethod3:(param1_type) parameter_varName;

-(return_type) instanceMethod1:(param1_type) param1_varName :(param2_type) param2_varName;
-(return_type) instanceMethod2WithParameter:(param1_type) param1_varName
andOtherParameter:(param2_type) param2_varName;
@end
```

method แบบ instance จะถูกนำหน้าด้วยเครื่องหมายลบ "-" ส่วน method แบบ class จะถูกนำหน้าโดยเครื่องหมายบวก "+" ตรงนี้จะแตกต่างจากUML diagramsซึ่งใช้ในแสดงว่า method เป็นแบบ private หรือ public

ค่าที่ถูก return จาก method มีลักษณะเช่นเดียวกับในภาษาซี เช่น void, int, ฯลฯ โดยจะมีการประกาศ type ชื่อ id ไว้แทน instance อ็อบเจกอะไรก็ได้

เราประกาศพารามิเตอร์ของ method ด้วยเครื่องหมายทวิภาคหรือโคลอน ":"ตามด้วย type ของพารามิเตอร์ในวงเล็บ แล้วตามด้วยชื่อของพารามิเตอร์เรามักใส่ชื่อที่มีความสอดคล้องกับพารามิเตอร์หน้าเครื่องหมายทวิภาคเพื่อบอกว่าพารามิเตอร์แต่ละตัวมีบทบาทอย่างไร

```
-(void) setRange: (int) start : (int) end;
-(void) importDocumentWithName: (NSString *) name withSpecifiedPreferences: (Preferences *) prefs beforePage: (int) insertPage;
```

@implementation

ส่วนอินเตอร์เฟซจะประกาศแต่ต้นแบบของ method เท่านั้น โดยไม่รวมถึงการกำหนดว่า method นั้นจะต้องทำอะไรบ้าง การกำหนดว่า method นั้นจะต้องทำอะไรจะทำในส่วนอิมพลีเมนเทชัน ตามปรกติส่วนอิมพลีเมนเทชันจะอยู่ในไฟล์นามสกุล .m ตัวอย่างเช่นThing.mที่มีการกำหนดส่วนอิมพลีเมนเทชันไว้ดังนี

```
@implementation classname
+ classMethod
{
   // implementation
}
- instanceMethod
{
   // implementation
}
@end
```

method มีหน้าตาแตกต่างจากฟังค์ชันในภาษา C เช่นถ้าในภาษาซีเป็นอย่างนี้

```
int do_something (int i)
{
   return square_root (i) ;
}
```

โดยมี int do_something (int) เป็นต้นแบบ

เมื่อมาประกาศเป็น method ในภาษาอ็อบเจ็กทีฟ-ซี จะมีหน้าตาอย่างนี้

```
- (int) do_something: (int) i
{
   return [self square_root: i];
}
```

ถ้าให้ถูกธรรมเนียมเราจะกำหนดชื่อ method ข้างต้นใหม่ โดยเราจะตั้งชื่อ method ให้สอดคล้องกับ argument ตัวแรกดังนี้

```
- (int) doSomethingWithInt: (int) i
{
   return [self squareRootOfInt:i];
}
```

แม้ว่าอาจจะดูยุ่งยากแต่ก็ช่วยให้เราจำ argument ได้ง่ายขึ้น ตัวอย่างเช่น

```
- (int) changeColorWithRed: (int) r  green: (int) g  blue: (int) b
```

โดยเราจะเรียก method แบบนี้:

```
[myColor changeColorWithRed:5 green:2 blue:6];
```

คอมไพเลอร์ภาษาอ็อพเจ็กทีฟ-ซีแต่ละตัวก็จะกำหนดชื่อ method เป็นการภายในแตกต่างกันไป เช่นถ้า method -changeColorWithRed:green:blue: เป็น method แบบ instance ของ class Color คอมไพเลอร์ก็อาจจำ method นี้ในชื่อ _i_Color_changeColorWithRed_green_blue เป็นต้น โดยตัว i จะบอกว่าจุดนี้คืออิมพลีเมนเทชัน method แบบ instance และตามด้วยชื่อคลาสและชื่อ method และมีการเปลี่ยนเครื่องหมาย : เป็น _ ดังนี้แล้ว เมื่อพารามิเตอร์เป็นส่วนหนึ่งของชื่อ method ลำดับของพารามิเตอร์ในภาษาอ็อบเจ็กทีฟ-ซีจึงสลับที่กันไม่ได้

อย่างไรก็ตาม เราแทบไม่ได้ใช้ชื่อภายในของฟังชั่นโดยตรง โดยทั่วไปการส่ง message จะถูกเปลี่ยนให้อยู่ในรูปแบบของการเรียกฟังค์ชั่นที่กำหนดไว้ในไลบราลีส่วน run-time และปรกติเวลาเราส่ง message เราก็จะไม่คำนึงถึง class ที่แท้จริงของส่วน receiver (myColor) อยู่แล้ว นั่นคือการจัดการหา method จะเป็นหน้าที่ของ run-time

2.6 XML

เอกซ์เอ็มแอล (อังกฤษ: XML) ย่อมาจาก Extensible Markup Language ซึ่งเป็นภาษามาร์กอัปสำหรับการใช้งานทั่วไป พัฒนาโดยW3C โดยมีจุดประสงค์เพื่อเป็นสิ่งที่เอาไว้ติดต่อกันในระบบที่มีความแตกต่างกัน (เช่นใช้คอมพิวเตอร์มี่มีระบบปฏิบัติการคนละตัวหรืออาจจะเป็นคนละโปรแกรมประยุกต์ที่มีความต้องการสื่อสารข้อมูลถึงกัน)นอกจากนี้ยังเพื่อเป็นพื้นฐานในการสร้างภาษามาร์กอัปเฉพาะทางอีกขั้นหนึ่ง XML พัฒนามาจาก SGML โดยดัดแปลงให้มีความซับซ้อนลดน้อยลง XML ใช้ในแลกเปลี่ยนข้อมูลระหว่างเครื่องคอมพิวเตอร์ที่แตกต่างกันและเน้นการแลกเปลี่ยนข้อมูลผ่านอินเทอร์เน็ต XML ยังเป็นภาษาพื้นฐานให้กับภาษาอื่น ๆ อีกด้วย (ยกตัวอย่างเช่น Geography Markup Language (GML) , RDF/XML, RSS, MathML, Physical Markup Language (PML) , XHTML, SVG, MusicXML และ cXML) ซึ่งอนุญาตให้โปรแกรมแก้ไขและทำงานกับเอกสารโดยไม่จำเป็นต้องมีความรู้ในภาษานั้นมาก่อน

2.6.1 Element

ประกอบไปด้วย แท็กเปิด ข้อมูล และแท็กปิด ยกตัวอย่างเช่น

```
<student> Example_sudent </student>
```

ในที่นี้<student>คือแท็กเปิด Example_sudent คือข้อมูล และ</student>คือแท็กปิดโดยแท็กปิดนั้นจะต้องมีชื่อเหมือนแท็กเปิดของมันแต่ตามหลังจากเครื่องหมาย '/' จะสังเกตได้ว่า XML นั้นคล้ายกับ HTML เป็นอย่างมากสำหรับข้อแตกต่างที่ชัดเจนคือ HTML ได้กำหนดแท็กไว้ล่วงหน้าแล้วแต่ XML ไม่ว่าใคร ๆ ก็สามารถกำหนดแท็กของเราเองได้ XML นั้นไม่ใช่ภาษาโดยสมบูรณ์มันเป็น*มาตรฐาน*ข้อมูลมากกว่าโดยตัวโปรแกรมประยุกต์จะเป็นผู้กำหนดรูปแบบของตัวเองขึ้นและจะสามารถใช้ได้กับโครงสร้างข้อมูลที่ถูกอนุญาต (เพราะว่ามีรูปแบบของข้อมูลที่เข้ากันได้) XML นั้นเป็นภาษาที่ case

sensitive ดังนั้นการที่เราเขียนว่า<student>กับ<Student>จึงถือว่าเป็นคนละแท็กกันนอกจากนี้แล้ว element ใน XML สามารถบรรจุอยู่ใน element อื่น ๆ ได้ยกตัวอย่างเช่น

```
<student>
<name>example name</name>
<id>123456789</id>
</student>
```

จะเห็นว่า element <name>บรรจุอยู่ภายใน element <student> element ไม่สามารถคร่อมกันได้ เช่น

```
<student>
<name>example name
<id></name>123456789</id>
</student>
```

แบบนี้ถือว่าไม่ถูกต้อง

แต่สามารถมี element ว่างแบบนี้ได้

```
<book></book>
```

นอกจากนี้ยังมีข้อยกเว้นสำหรับแท็กว่างจะเป็นแท็กที่ไม่ต้องมีแท็กปิดได้โดยสามารถเขียน

```
<book />
```

ซึ่งมีความหมายเหมือนกับด้านบน

2.6.2 Attribute

นอกจากแท็กแล้วยังมี สิ่งที่เรียกว่า attribute ด้วยโดยที่มีรูปแบบดังนี้

```
<student name="example_name"></student>
<student name='example_name'></student>
```

จะเห็นว่าทั้งสองแบบมีความเหมือนกันแตกต่างกันเล็กน้อยคือใช้เครื่องหมาย " กับ ' ซึ่งสามารถใช้ได้ทั้งคู่

2.6.3 การประกาศ XML

ส่วนต่าง ๆ ของ XML คือ node

node ที่ปรากฏบน XML ทุกฉบับคือ การประกาศ Declaration ซึ่งมีลักษณะเหมือน แท็กแต่มีเครื่องหมาย ? อยู่ด้วย โดยในเอกสาร XML จะต้องมี node นี้ทุกฉบับ

```
<?xml version="1.0" ?>
```

2.6.4 โครงสร้างของเอกสาร XML

ถูกกำหนดขึ้นโดยลำดับชั้น โดยเอกสารใด ๆ นั้นต้องมี root element หนึ่งตัวเสมอ เช่นในที่นี้คือ

```
<student>
<?xml version="1.0" ?>
<student>
<name>example name</name>
<id>123456789</id>
</student>
```

2.6.5 การตรวจสอบความถูกต้องของ XML

ความถูกต้องของ XML แบ่งเป็น 2 ระดับ

1. **Well-formed**เอกสารที่ well-formed คือใช้ syntax ของ XML ถูกต้องตามมาตรฐานทุกอย่าง มีการเปิด-ปิดแท็กที่สมบูรณ์แต่ไม่จำเป็นต้องจัดรูปแบบให้สวยงาม เอกสารที่ไม่ well-formed ถือว่าไม่เป็น XML
2. **Valid**นอกจาก well-formed แล้ว เอกสารที่ valid ยังต้องใช้แท็ก XML ที่กำหนดเฉพาะใน schema ที่ตกลงกันไว้เท่านั้น ปัจจุบันมี schema ที่นิยม 3 ตัว คือ Document Type Definition
3. (DTD) , XML Schema (W3C) (WXS) และ RELAX NG

2.7 SQL

2.7.1 **SQL SELECT**

เป็นคำสั่งที่ใช้สำหรับการเรียกดูข้อมูลในตาราง (**Table**) คำสั่ง**SQL SELECT**สามารถเรียกได้ทั้งตาราง หรือว่า สามารถระบุฟิวด์ที่ต้องการเรียกดูข้อมูลได้

```
SELECT Column1, Column2, Column3,... FROM [Table-Name]
```

2.7.2 **SQL WHERE**

เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง (**Table**) คำสั่ง**SQL WHERE** สามารถระบุเงื่อนไขในการเลือกข้อมูลได้ 1 เงื่อนไข หรือมากกว่า 1 เงื่อนไข

```
SELECT Column1, Column2, Column3,... FROM Table-Name WHERE [Field] = 'Value'
```

2.7.3 **SQL OR AND**

เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง(**Table**) การเชื่อมวลีสำหรับเงื่อนไขต่าง ๆ

```
SELECT Column1,Column2,Column3,... FROM [Table-Name] WHERE [Field] = 'Value'
[AND/OR] [Field] = 'Value'
```

2.7.4 **SQL ORDER BY**

เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง (**Table**) โดยจัดเรียงข้อมูลตามต้องการ

```
SELECT Culumn1,Culumn2,Culumn3,... FROM [Table-Name] ORDER BY [Field]
[ASC/DESC],[Field] [ASC/DESC],...
ASC = น้อยไปหามาก
DESC = มากไปหาน้อย
```

2.7.5 **SQL SUB SELECT QUERY**

เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง (**Table**) โดยใช้เลือกข้อมูลย่อยภายใน SELECT ย่อยอีกชั้นหนึ่งครับ**SUB SELECT QUERY**เข้ามาช่วยในด้านความสะดวกและง่ายกว่าการ**JOIN TABLE**แต่ข้อเสียของ**SUB SELECT** คือ สามารถทำงานได้ช้ากว่า**JOIN TABLE**

```
SELECT Column1,Column2,Column3,... FROM [Table-Name] WHERE [Field] IN (SELECT .....
FROM ....)
```

2.7.6 **SQL SELECT INTO**

เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง (**Table**) โดยใช้การเลือกข้อมูลจากต้นทางไปยังปลายทาง นิยมใช้สำหรับการ**Copy Table**หรือทำการ**Backup Table**

```
SELECT Column1,Column2,Column3,... INTO [New-Table] FROM [Table-Name]
```

2.7.7 **SQL JOIN**

เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง (**Table**) โดยเงื่อนไขการ JOIN จะกระทำเมื่อมีข้อมูลตั้งแต่ 2 Table ขึ้นไปโดยข้อมูลเหล่านั้นเป็นข้อมูลที่มีความสัมพันธ์และเชื่อมโยงกับข้อมูลหลัก

```
SELECT [Table-Name1].Column1, [Table-Name2].Column1,... FROM [Table-Name1],[Table-
Name2]
WHERE [Table-Name1].Column = [Table-Name2].Column
```

2.7.8 **SQL DISTINCT**

เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง (**Table**) โดยทำการเลือกข้อมูลที่ซ้ำกันมาเพียงแค่ Record เดียว

```
SELECT DISTINCT Column1,Column2,Column3,... FROM [Table-Name]
```

2.7.9 **SQL LIKE**

เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง (**Table**) โดยทำการค้นหาข้อความที่ระบุภายในฟิวด์ที่กำหนด

```
SELECT Column1,Column2,Column3,... FROM [Table-Name] WHERE [Filed] LIKE '%Value%'
```

2.7.10 **SQL NOT LIKE**

เป็นคำสั่งที่ใช้สำหรับการระบุเงื่อนไขการเลือกข้อมูลในตาราง (**Table**) โดยทำการค้นหาข้อความที่ระบุภายในฟิวด์ที่กำหนด และไม่แสดง Record ที่ค้นพบ ซึ่งทำหน้าที่ตรงข้ามกับ**LIKE**

```
SELECT Column1,Column2,Column3,... FROM [Table-Name] WHERE [Filed] NOT LIKE
'%Value%'
```

2.7.11 **SQL INSERT**

เป็นคำสั่งที่ใช้สำหรับเพิ่มข้อมูลลงในตาราง (**Table**) โดยสามารถเพิ่มได้ทั้งแถวหรือว่าเพิ่มในส่วนของแต่ละฟิวด์

```
INSERT INTO [Table-Name] (Column1,Column2,Column3,...) VALUES
('Value1','Value2','Value3',...)
```

2.7.12 **SQL INSERT ... SELECT**

เป็นคำสั่งที่ใช้สำหรับเพิ่มข้อมูลลงในตาราง (**Table**) โดยสามารถทำการ**เพิ่มข้อมูล**โดยการ SELECT ข้อมูลจาก Table (ตาราง) อื่น ๆ

```
INSERT INTO [Table-Name] (Column1,Column2,Column3,...) SELECT
(Column1,Column2,Column3,...) WHERE ...
```

2.7.13 **SQL UPDATE**

เป็นคำสั่งที่ใช้สำหรับแก้ไขข้อมูลในตาราง (**Table**) โดยสามารถทำการแก้ไขได้หลายฟิวด์และหลาย Record ภายในคำสั่ง 1 คำสั่ง ทั้งนี้ขึ้นอยู่กับ Where ที่ผู้ใช้ได้เขียนขึ้น

```
UPDATE [Table-Name] SET Column1='Value1',Column2='Value2',... WHERE clause
```

2.8 Graphic User Interface (GUI)

User Interface(UI) สิ่งที่มีไว้ให้ผู้ใช้ใช้ในการกระทำกับระบบหรือสิ่งของต่าง ๆ ซึ่งอาจจะเป็นคอมพิวเตอร์เครื่องจักรเครื่องกลอุปกรณ์ใช้ไฟฟ้าใด ๆ หรือระบบที่มีความซับซ้อนอื่น ๆ เพื่อให้สิ่ง ๆ นั้นทำงานตามความต้องการของผู้ใช้

ส่วนต่อประสานกับผู้ใช้สามารถจัดได้เป็น 2 ประเภทใหญ่ได้แก่

- ส่วนที่นำข้อมูลเข้า หรือส่วนสั่งงาน เรียกว่า input
- ส่วนที่ใช้แสดงผลลัพธ์ หรือส่วนที่ไว้รอคำสั่งจากผู้ใช้ เรียกว่า output

ตัวอย่างของส่วนต่อประสานกับผู้ใช้ เช่น เครื่องคิดเลข จะมีส่วน Input คือแป้นตัวเลข 0-9 และเครื่องหมายทางคณิตศาสตร์ ผู้ใช้จะต้องกดหมายเลขที่ต้องการคำนวณผลผ่านแป้นตัวเลขนั้น และเมื่อเครื่องคิดเลขทำการประมวลผลเสร็จสิ้น ก็จะแสดงผลลัพธ์ออกมาบนหน้าจอ LED ซึ่งเป็นส่วนของ Output

แต่ในส่วนของ GUI จะเน้นไปทาง Graphic ที่แสดงออกมาจาก iPhone ให้กับผู้ใช้ผ่านทางสัญลักษณ์หรือภาพนอกเหนือจากทางตัวอักษร โดยมีหลักการเหมือนกับ UI คือมีทั้ง Input และ Output เช่น สัญลักษณ์ ภาพ และส่วนที่เพิ่มขึ้นมา คือ ไอคอน หน้าต่างการใช้งาน เมนู ปุ่มเลือก และการใช้เมาส์ หรือแม้แต่ในระบบ Touch screen

ในเรื่องของส่วนต่อประสานกับผู้ใช้ มีศาสตร์หนึ่งที่เป็นส่วนระบุถึงคุณภาพของส่วนติดต่อผู้ใช้เรียกว่า ความเหมาะสมต่อการใช้งาน หรือ Usability

เป็นศาสตร์ที่ว่าด้วยการออกแบบส่วนต่อประสานกับผู้ใช้ให้มีความเหมาะสมต่อการใช้งาน กล่าวคือสามารถใช้งานได้ง่าย สะดวก รวดเร็ว โดยศาสตร์ของความเหมาะสมต่อการใช้งาน จะมีความเกี่ยวข้องกับด้าน จิตวิทยา และ สรีรวิทยา เป็นหลัก ซึ่งจิตวิทยาจะช่วยให้นักออกแบบส่วนติดต่อผู้ใช้สามารถคิดระบบระเบียบขั้นตอนวิธีการใช้งานที่ทำให้ผู้ใช้สามารถใช้งานอุปกรณ์ได้สะดวก และง่ายต่อการทำความเข้าใจ ส่วนสรีรวิทยาจะช่วยในการออกแบบให้อุปกรณ์นั้นเหมาะสมต่อกับใช้ในด้านสรีระ เช่น ความสะดวกในการ Touch จอ เป็นต้น

ส่วนต่อประสานกราฟิกกับผู้ใช้ ช่วยให้การเรียนรู้การใช้งานคอมพิวเตอรทำได้ง่ายและเร็วขึ้นเปรียบเทียบกับระบบเก่า

ที่ต้องพิมพ์ชุดคำสั่งที่ใช้ในระบบดอสหรือยูนิกซ์ ซึ่งเป็นส่วนต่อประสานผู้ใช้แบบข้อความ (Text user interface) ในบางครั้งจะเรียกว่า ส่วนต่อประสานแบบชุดคำสั่ง (Command Line Interfaces, CLI)

ในการใช้งานผู้ใช้ต้องพิมพ์คำสั่งผ่านทางคีย์บอร์ด โดยคำสั่งเฉพาะต่าง ๆ ที่พิมพ์จำเป็นต้องใช้เวลาระยะหนึ่งในการเรียนรู้ อย่างไรก็ดีในบางระบบเช่น Linux ก็ยังใช้ GUI เป็น frontend เพื่อที่ทำงาน กับส่วนต่อประสานแบบชุดคำสั่ง รวมถึงการพัฒนารูปแบบของ GUI ได้มีการพัฒนาไปอย่างรวดเร็วโดยมีทั้งในส่วนของ Commercial และ แบบ โอเพ่นซอร์ส ซึ่งในส่วนของ Commercial ได้มีบริษัทยักษ์ใหญ่บางบริษัทเริ่มเข้าจับตลาดทางด้านนี้แล้วเช่น Sun ก็มีในส่วนของ Java Desktop เป็นต้น โดยกล่าวถึงรูปแบบการติดต่อกับผู้ใช้แบบนี้ว่า Desktop ระบบของ GUI เช่นในส่วนของระบบปฏิบัติการวินโดว์จะเป็นในลักษณะถูกสร้างขึ้นมาให้เป็นส่วนหนึ่งกับตัวระบบปฏิบัติการเลย โดยมีการเรียก ใช้ส่วนการติดต่อกับผู้ใช้ผ่านทางระบบ API โดยไปเรียกส่วนประมวณผลที่ชื่อว่า GDI และมีโหมดที่เป็นส่วนติดต่อกับผู้ใช้แบบชุดคำสั่ง แยกเป็นส่วนหนึ่งอีกต่างหากแต่ยังคงอยู่ภายใต้ระบบปฏิบัติการ (shell) ระบบ GUI ในส่วนของ Opensource ได้มีการจัดตั้งองค์การขึ้นมาเพื่อทำการกำหนดมาตรฐานกลางที่ใช้ในทำงานในส่วนการพัฒนาของ Desktop ร่วมกันเช่น การกำหนดมาตรฐานของเมนู, ลักษณะส่วนติดต่อย่อยอื่น ๆ เป็นต้นโดยองค์นี้มีชื่อว่า freedesktop Desktop ที่ร่วมใช้มาตรฐานเดียวกันกับ freedesktop เช่น Gnome,KDE,XFCE เป็นต้น รวมถึงมีการพัฒนาลูกเล่นต่าง ๆ ออกไปมากมาย รวมถึงการใช้งานที่หลากหลายมากขึ้นโดยลดการติดต่อระหว่าง ผู้ใช้และส่วนที่เป็นการประสานชุดคำสั่งลงให้มากที่สุด ในระบบปฏิบัติการใหม่ ๆ ในปัจจุบันมีการรองรับการใช้งานทั้งแบบกราฟิกและแบบข้อความ โดยแสดงผลผ่านทางกราฟิกเป็นหลักสำหรับผู้ใช้ทั่วไป และต้องใช้คำสั่งพิเศษเพื่อเรียกใช้คำสั่งของการใช้ส่วนต่อประสานแบบข้อความ โดยแบ่งแยกออกเป็น โหมดการทำงานได้ด้วย

2.9 Object-oriented programming (OOP)

คือวิธีการหนึ่งในการเขียนโปรแกรมในเชิงวัตถุ ซึ่งสามารถทำงานร่วมกัน หรือนำมารวมกันได้โดย การแลกเปลี่ยนข่าวสาร เพื่อใช้ในการประมวลผลและส่งข่าวสารที่ได้รับไปให้วัตถุอื่น ๆ ที่เกี่ยวข้อง เพื่อทำงานต่อ ซึ่งจะให้ความสำคัญ กับ ข้อมูล(Data)และ พฤติกรรม(Behavior)ของวัตถุและความสัมพันธ์กันระหว่างวัตถุ

2.10 การสร้างโมเดลความสัมพันธ์ระหว่างข้อมูล : ER – DIAGRAM

แนวคิดเกี่ยวกับ ER-DIAGRAM

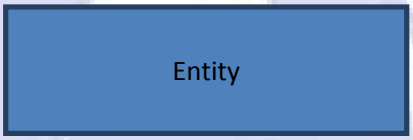
ER-DIAGRAM ประกอบด้วยองค์ประกอบพื้นฐานดังนี้

- เอนทิตี้ (Entity) เป็นวัตถุ หรือสิ่งของที่เราสนใจในระบบงานนั้น
- แอททริบิว (Attribute) เป็นคุณสมบัติของวัตถุที่เราสนใจ
- ความสัมพันธ์ (Relationship) คือ ความสัมพันธ์ระหว่างเอนทิตี้

เอนทิตี้ (Entity)

เอนทิตี้ หมายถึง สิ่งของหรือวัตถุที่เราสนใจ ซึ่งอาจจับต้องได้ และเป็นได้ทั้งนามธรรม โดยทั่วไปเอนทิตี้จะมีลักษณะที่แยกออกจากกันไป เช่น เอนทิตี้ของพนักงาน จะแยกออกเป็นพนักงานเลย เอนทิตี้เงินเดือนของพนักงานคนหนึ่งก็อาจเป็นเอนทิตี้หนึ่งในระบบของโรงงาน

โดยทั่วไปแล้ว เอนทิตี้จะมีกลุ่มที่บอกคุณสมบัติที่บอกลักษณะของเอนทิตี้ เช่น พนักงานมีรหัส ชื่อ นามสกุล และแผนก โดยจะมีค่าของคุณสมบัติบางกลุ่มที่ทำให้สามารถแยกเอนทิตี้ออกจากเอนทิตี้ เช่น รหัสพนักงานที่จะไม่มีพนักงานคนไหนใช้ซ้ำกันเลย เราเรียกว่าคุณสมบัติกลุ่มนี้ว่าเป็นคีย์ของเอนทิตี้



ภาพที่ 2.25แสดงสัญลักณ์ของเอนทิตี้

แอททริบิวท์ (Attribute)

Attribute คือ คุณสมบัติของวัตถุหรือสิ่งของที่เราสนใจ โดยอธิบายรายละเอียดต่าง ๆ ที่เกี่ยวของกับลักษณะของเอนทิตี้โดยคุณสมบัตินี้มีอยู่ในทุกเอนทิตี้ เช่น ชื่อ นามสกุล ที่อยู่ แผนก เป็น Attribute ของเอนทิตี้พนักงาน

โดยทั่วไปแล้วโมเดลข้อมูล เรามักจะพบว่า Attribute มีลักษณะข้อมูลพื้นฐานอยู่โดยที่ไม่ต้องมีคำอธิบายมากมาย และ Attribute ก็ไม่สามารถอยู่แบบ โดด ๆ ได้โดยไม่มีเอนทิตี้

ภาพที่ 2.26 แสดงสัญลักณ์ของแอททริบิวท์

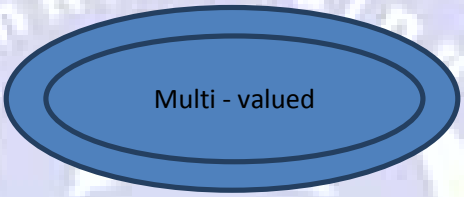
ชนิดของ Attribute สามารถแบ่งออกได้หลายลักษณะดังนี้

Simple และ Composite

- Simple Attribute คือ Attribute ที่ไม่สามารถแยกออกเป็นส่วนย่อยได้เช่นรหัส
- Composite Attribute คือ Attribute ที่สามารถแยกออกเป็นส่วนย่อยได้ เช่น ชื่อ อาจจะประกอบด้วยชื่อต้น และนามสกุล

Single – valued และ Multi – valued atttribute

- Single – valued คือ ค่าของเอนทิตี้ที่สามารถมีได้แค่ค่าเดียว เช่น วันเกิด สำหรับพนักงานแล้วสามารถมีได้เพียงค่าเดียวจึงให้สัญลักษณ์ของ Attribute ปกติ
- Multi – valued คือ ค่าที่เป็นไปได้มากว่า 1 ค่า เช่น ทำเลที่ตั้งของโรงงานสามารถมีได้มากว่า 1 แห่ง



ภาพที่ 2.27 แสดงสัญลักษณ์ของ Multi - valued

Stored และ Derived attribute

- Stored Attribute จะเป็น Attribute ที่เก็บอยู่ในฐานข้อมูลเช่น วันเกิด ใช้สัญลักณ์ปกติ
- Derived Attribute เป็น Attribute ที่เกิดจากการคำนวณ เช่น อายุ เกิดจากการคำนวณวันเกิดกับช่วงเวลาปัจจุบัน

ภาพที่ 2.28 แสดงสัญลักษณ์ของ Derived attribute

ความสัมพันธ์ (Relationship)

เอนทิตี้แต่ละเอนทิตี้ต้องมีความสัมพันธ์ (Relationships Degree) จะบอกถึงความสัมพันธ์ระหว่างเอนทิตี้ มีดังนี้

- ความสัมพันธ์เอนทิตี้เดียว (Unary Relationships) หมายถึง เอนทิตี้หนึ่ง ๆ จะมีความสัมพันธ์กับตัวมันเอง
- ความสัมพันธ์สองเอนทิตี้ (Binary Relationships) หมายถึง เอนทิตี้สองเอนทิตี้จะมีความสัมพันธ์กัน
- ความสัมพันธ์สามเอนทิตี้ (Ternary Relationships) หมายถึง เอนทิตี้สามเอนทิตี้มีความสัมพันธ์กัน

การระบุตำแหน่งความสัมพันธ์ระหว่างเอนทิตี้ (Connectivity)

การระบุตำแหน่งความสัมพันธ์ระหว่างเอนทิตี้ (Connectivity) ว่าเป็นแบบหนึ่งต่อหนึ่ง (One to One Relationships), แบบหนึ่งต่อกลุ่ม (One to Many Relationships) หรือ แบบกลุ่มต่อกลุ่ม (Many to Many Relationships) นั้นจะใช้ Connectivity เพื่อระบุตำแหน่ง 1, M หรือ N ไว้ข้างใดของเอนทิตี้

คีย์ (Key)

- Super key : Attribute หรือกลุ่มของ Attribute ซึ่งมีค่าแตกต่างกันไปในแต่ล่ะเอนทิตี้สามารถระบุเอนทิตี้เฉพาะตัวหนึ่ง ๆ ได้
- Candidate Key : Subset ที่เล็กที่สุดของ Super key ที่สามารถระบุเฉพาะเอนทิตี้นั้นได้
- Primary Key : Candidate Key ที่ถูกเลือกให้เป็นตัวระบุหรือ identity เอนทิตี้เฉพาะตัว

การแปลง ER Model ให้อยู่ในรูปแบบโครงสร้างฐานข้อมูล

1. แปลงเอนทิตี้ที่ความสัมพันธ์แบบหนึ่งต่อหนึ่ง (One to One Relationships) ไปเป็นตาราง โดยแทนที่หนึ่งเอนทิตี้เป็นหนึ่งตาราง Attribute แต่ละเอนทิตี้เป็นฟิลด์หรือคอลัมน์แต่ละตาราง
2. แปลงเอนทิตี้ที่มีความสัมพันธ์แบบหนึ่งต่อกลุ่ม (One to Many Relationships) ไปเป็นตาราง โดยด้านเอนทิตี้ที่เป็นตัวเลข 1 นั้นสามารถแปลงเป็นตารางได้ทันที Attribute ของเอนทิตี้นั้นจะเป็นฟิลด์ของตารางทันที ส่วนด้านเอนทิตี้เป็นตัวอักษร M แสดงเอนทิตี้เป็น

ตารางโดย Attribute ของเอนทิตี้ตัวมันเองและนำคีย์หลักของเอนทิตี้ที่เป็นเลข 1 มาใส่ฟิลด์ของตารางนั้นด้วย

3. แปลงเอนทิตี้ที่มีความสัมพันธ์แบบกลุ่มต่อกลุ่ม (Many to Many Relationships) ไปเป็นตารางโดยสร้างเอนทิตี้กลาง (Composite Entity) เอนทิตี้กลางจะนำคีย์หลัก ๆ ของทั้งสองตารางมาเป็นคีย์หลักของเอนทิตี้หลักของเอนทิตี้กลางด้วยส่วนเอนทิตี้ทั้งสองที่อยู่ระหว่างเอนทิตี้กลางก็แปลงเป็นตารางได้ โดยนำเอา Attribute ของแต่ล่ะเอนทิตี้ไปเป็นฟิลด์ (ทำตามกฎของงความสัมพันธ์แบบหนึ่งต่อหนึ่ง)

2.11 การนอร์มัลไลซ์ (Normalization)

2.11.1 ความหมายของคำที่เกี่ยวข้อง

2.11.1.1FD (Functionally Dependent หรือ Function Dependency)

เป็นความสัมพันธ์ระหว่าง attribute แบบ m:1 หรือ 1:1 โดย attribute ทางขวามีฟังก์ชันขึ้นกับ attribute ทางซ้าย

กรณี 1:1 ค่าของ attribute ทางซ้าย 1 ค่า จะสัมพันธ์กับค่าของ attribute ทางขวา 1 ค่า

กรณี m:1 ค่าของ attribute ทางซ้ายมากกว่า 1 ค่าจะสัมพันธ์กับค่าของ attribute ทางขวา 1 ค่า

นิยาม

“กำหนดให้ x และ y เป็น attribute ของ relation R จะได้ว่า y มีฟังก์ชันขึ้นกับ x (y เป็น FD กับ x) ก็ต่อเมื่อถ้า 2tuples ใน R มีค่าของ x ตรงกันแล้ว ทั้ง 2 tuples จะต้องมีค่าของ y ตรงกันด้วย”

ถ้า relation R มี attribute x, y จะได้ว่า

Rx -> Ry

อ่านว่า x determine y หรือ y depends on x

หมายความว่า x จะเป็นตัวกำหนดค่า (determine) ของ y หรือค่าของ y จะขึ้นอยู่กับค่าของ x (y depends on x หรือ y เป็น FD กับ x) เราจะเรียกว่า determinant

หลักในการเขียน Functional Dependency

FDs : determinant-attribute -> dependency-attribute

FDs : PRESON_ID -> PERSON_NAME

ประเภทของ Functional Dependency

- FD ที่เกิดจากความสัมพันธ์ระหว่าง Determinant และ Dependency อย่างละ 1 ค่า
  PERSON_ID -> PERSON_NAME
- FD ที่เกิดจากความสัมพันธ์ระหว่าง Determinant 1 ค่า กับ Dependency หลายค่า
  PERSON_ID -> FNAME, LNAME, ADDRESS, BIRTH_DATE, ISSUE_DATE
- FD ที่มีความสัมพันธ์ 2 ทาง ที่ทั้ง Determinant และ Dependency ต่างสามารถทำหน้าที่ของอีกฝ่ายหนึ่งได้

  PROJECT_NO -> MANAGER

  MANAGER -> PROJECT_NO

  PROJECT_NO <- -> MANAGER
- FD ที่ต้องใช้ Determinant มากว่า 1 ค่าเพื่ออ้างถึง Dependency PRODUCT_LINE, ITEM_NO -> USED_QTY

2.11.1.2 Full FD (Full Functional Dependence)

นิยาม

“attribute y ของ releation R จะเป็น Full FD บน attribute x ของ relation R ถ้า y เป็น FD กับ x และไม่เป็น FD กับ Subset ใด ๆ ของ x”

หรืออาจกล่าวได้ว่าเป็น FD ที่มี Determinant ขนาดเล็กที่สุด และสามารถระบุถึง Dependency ได้ ดังตัวอย่างต่อไปนี้

- D1 : PERSON_ID -> ADDRESS
- D2 : PRESON_ID, PERSON_NAME -> ADDRESS
- D3 : PRODUCT_LINE, ITEM_NO -> USE_QTY
- D4 : PRODUCT_LINE, ITEM_NO, MANAGER -> USE_QTY

ดังนั้นจากตัวอย่างสามารถสรุปได้ว่า D1 กับ D3 ถือเป็น Full FD

2.11.1.3Partial Dependency

ความสัมพันธ์แบบนี้จะเกิดขึ้นได้ก็ต่อเมื่อ relation หนึ่ง ๆ มีคีย์หลักเป็นคีย์ผสม (composite key) นั่นคือ คีย์หลักของ relation นั้น ๆ ประกอบด้วย attribute หลาย attribute ความสัมพันธ์ระหว่างค่าของ

attribute แบบบางส่วนเกิดขึ้นเมื่อ attribute บางส่วนของคีย์หลักสามารถระบุค่าของ attribute อื่น ๆ ที่ไม่ใช่คีย์หลักของ relation ได้ (Non-key attribute) ซึ่งความสัมพันธ์แบบนี้จะทำให้เกิดปัญหาในเรื่องของความซ้ำซ้อน และการปรับปรุงข้อมูล

2.11.1.4 Transitive Dependency

Attribute ที่มีคุณสมบัติเป็นคีย์หลักจะสามารถระบุค่าของทุก attribute ในแต่ละ tuples ได้อย่างไรก็ตาม ในบาง relation อาจจะมีกรณี attribute อื่น (Non-key attribute) ที่สามารถระบุค่าของ attribute อื่น ๆ ใน tuples ได้ ลักษณะของความสัมพันธ์ในการระบุค่า attribute แบบนี้เรียกว่า ความสัมพันธ์ระหว่างค่าของ attibute แบบนี้ว่า Transitive Dependency ดังตัวอย่าง

รหัสนักศึกษา ->รหัสสาขาวิชา

รหัสสาขาวิชา ->ชื่อสาขาวิชา

จะเห็นได้ว่า รหัสสาขาวิชา ->ชื่อสาขาวิชา เป็น Transitive Dependency ซึ่งถือว่าเป็นส่วนที่เกิดมาโดยไม่จำเป็น เนื่องจาก รหัสนักศึกษา ->รหัสสาขาวิชา ก็สามารถดึงข้อมูลในส่วนของชื่อสาขาวิชาได้ ดังนั้น เราจะไม่เก็บชื่อสาขาวิชาไว้ในตารางนักศึกษา

2.11.1.5 Multivalued Dependency

การขึ้นต่อกับหลายค่า (Nultivalued Dependency) ในความสัมพันธ์ R มี attribute A, B, C เราจะกล่าวว่า attribute B เป็นการขึ้นต่อกันหลายค่าบน A ก็ต่อเมื่อเซทของค่า attribute B ในความสัมพันธ์ R ที่มีความสัมพันธ์ตรงกับคู่ของ <A, C>นั้น จะไม่ขึ้นกับค่าของ C เขียนแทนด้วย A ->> B และจะมีลักษณะที่เป็นได้ดังนี้

- attribute A ค่าหนึ่งจะเป็นตัวกำหนดกลุ่มของค่า attribute B คือ เมื่อ 2tuples ในความสัมพันธ์ R มีค่า A เดียวกัน ไม่จำเป็นต้องมีค่า B เหมือนกัน แต่ค่าของ B จะต้องอยู่ในกลุ่มของค่า B ที่ถูกกำหนดโดย A
- การเปลี่ยนแปลค่าใดใน attribute C จะไม่มีผลกระทบต่อค่า B
- สอง tuples ในความสัมพันธ์ R ที่มีค่า B เหมือนกันไม่จำเป็นต้องมีค่า A เดียวกัน
- ค่าของ atturbute C สองค่าที่มีความสัมพันธ์กับค่า A เดียวกันจะต้องสัมพันธ์กับค่าของ B ในกลุ่มเดียวกันและ เป็นกลุ่มที่ถูกกำหนดโดยค่า A นั้น ๆ

หรืออาจกล่าวได้ว่า ค่าของ Determinant 1 ค่าสามารถระบุค่าของ attrubute ที่ทำหน้าที่เป็น Dependency ได้ตั้งแต่ 2 attrubute ขึ้นไป ซึ่งอยู่ในรูปของชุดข้อมูล

EMPLOYEE# -> -> DEPARTMENT#, PROJECT#

2.11.1.6 Trivial FD's

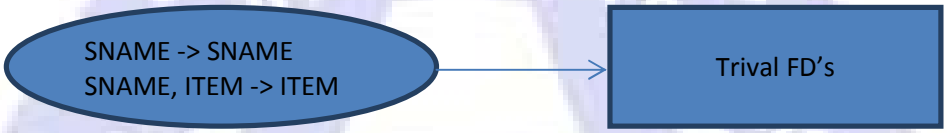
FD (Function Dependancy) เป็น Tribial FD's ก็ต่อเมื่อ attribute ทางด้านขวาอยู่ใน attribute ทางด้ายซ้าย

ตัวอย่างเช่น

SUPPLIER(SNAME, ADDRESS, ITEM, PRICE)

SNAME -> ADDRESS

SNAME, ITEM -> PRICE



2.11.2 การนอร์มัลไลซ์ (Normalization)

หลังจากที่ผู้ออกแบบได้ขอบเขตข้อมูลทั้งหมดที่ต้องการเก็บแล้ว ซึ่งโดยมากเกิดจากรูปแบบรายงานบ้าง รูปแบบในเสร็จรับเงินบ้าง รูปแบบใบส่งสินค้าบ้าง โดยมากมักจะเหมารวมเอาว่านั้นคือรูปแบบของตารางที่ต้องการเก็บข้อมูล ซึ่งนำมาซึ่งความซ้ำซ้อนของข้อมูลในหลายกรณี และทำให้มีส่วนซ้ำซ้อนขนาดใหญ่เกิดความจำเป็น ปัญหาของรีเลชั่นที่เกิดขึ้นเหล่านี้ สามารถขจัดได้ด้วย "ขบวนการ Normalization" ซึ่งแนวคิดนี้ถูกคิดค้นโดย E.F.Codd ซึ่งเป็นกระบวนการที่นำเค้าร่างของ relation มาทำให้อยู่ในรูปแบบเป็นบรรทัดฐาน (Normal Form) เพื่อให้แน่ใจว่าการออกแบบเค้าร่างของ relation เป็นการออบแบบที่เหมาะสม

ให้ทำการตรวจสอบเอนทิตีต่าง ๆ ให้อยู่ในกฎนอร์มัลไลเซชั่นซึ่งประกอบด้วย 1NF, 2NF, 3NF, BCNF, 4NF, 5NF ซึ่งจะได้กล่าวในหัวข้อต่อไป ประโยชน์ก็คือ

1. ลดที่ว่างที่ต้องใช้ในการเก็บข้อมูล
2. ลดความผิดพลาด ความไม่ตรงกันของข้อมูลในฐานข้อมูล
3. ลดการเกิดอะนอร์มัลไลของการลบและแก้ไขข้อมูล

4. เพื่อความคงทนแก่โครงสร้างฐานข้อมูล

ในทางปฏิบัติการทำนอร์มอลไลซ์จะเริ่มจาก ER Model ก่อน แล้วจึงทำการ map จาก ER model เป็น relation แบบ 1NF ก่อน โดยให้ attributes ที่เกี่ยวข้องกันจะอยู่ในตารางเดียวกันสำหรับ application ใหญ่ ๆ มี attributes ประมาณ 500 attributes ใช้ ER model จะได้ 1NF ประมาณ 80 ตาราง เมื่อทำถึง 5NF จะได้ไม่เกิน 100 ตาราง ในกรณีได้ตารางเป็นนอมัลไลเซชั่นที่สมบูรณ์แล้ว สิ่งที่ต้องระวังคือไม่แตกตารางนั้นย่อยลงไปอีก

ระดับนอร์มัลไลเซชั่น

นอร์มัลไลเซชั่นเป็นกระขวนการเพื่อพัฒนาการ เชื่อต่อของข้อมูลเพื่อแก้ปัญหาของรีเลชั่นที่ว่าการออกแบบฐานข้อมูลทั้งทรางตรรกะ และทางกายภาพที่ได้ออกมาใช้ได้หรือยัง การนอร์ลัลไลเซชั่นแบ่งออบเป็นหลายระดับได้แก่

2.11.2.1 First Normal Fore (1NF)

ทุก ๆ field ในแต่ละ reconrd จะเป็น single value นั้นคือ ในตารางหนึ่ง ๆ จะไม่มีค่าของกลุ่มข้อมูลที่ซ้ำกัน (Repeating Group) ตัวอย่างเช่น ตารางดังต่อไปนี้

ตารางที่ 2.1 แสดงลักษณะข้อมูลเป็น Repeating group

รหัสนักศึกษา	ชื่อ	นามสกุล	รหัสวิชาที่ลงทะเบียน
001	สมชาย	สมใจนึก	204-101
			204-204
			204-205
002	ธีรชาย	บุญมาศ	204-102
			204-204

เราสามารถทำให้อยู่ในรูป 1NF ได้ดังนี้

ตารางที่ 2.2 แสดงลักษณะข้อมูลเป็น 1NF

รหัสนักศึกษา	ชื่อ	นามสกุล	รหัสวิชาที่ลงทะเบียน

001	สมชาย	สมใจนึก	204-101
001	สมชาย	สมใจนึก	204-204
001	สมชาย	สมใจนึก	204-205
002	ธีรชาย	บุญมาศ	204-102
002	ธีรชาย	บุญมาศ	204-204

จะเห็นว่าการเก็บข้อมูลแบบนี้เป็นการสิ้นเปลืองโดยใช่เหตุเพราะมีค่าของกลุ่มข้อมูลที่ซ้ำกันมากมาย เพราะนักศึกษาคนหนึ่ง สามารถลงทะเบียนได้มากว่าหนึ่งวิชา

สรุปก็คือ นอร์มัลไลเซชั่นระดับที่ 1 (First normal form : 1NF) เป็นการขจัดแอตตรีบิว หรือกลุ่มแอตตริบิวที่ซ้ำกันไปอยู่ในแอนทิตีลูก เพื่อแต่ละรายการในเอนทิตี ไม่มีค่าของแอตตริบิว หรือค่าของกลุ่มแอตตริบิวที่ซ้ำกัน

สำหรับ 1NF จะมีข้อเสียในการแก้ไข การลบ และการเพื่อข้อมูล ดังนี้

1) การแก้ไขข้อมูล (Update) เนื่องจากมีข้อมูลอยู่หลาย tuples จะต้องแก้ไขทุก tuples นั้นคือต้องมีการแก้ไขข้อมูลมากกว่าหนึ่งแห่ง
2) การลบข้อมูล (Delete) ถ้าต้องการลบข้อมูลบางส่วนออกไป จะทำให้ลบข้อมูลอื่น ๆ ออกไปด้วยโดยไม่ตั้งใจ
3) การเพิ่มข้อมูล (Insert) อาจจะทำให้ไม่สามารถเพิ่มข้อมูลบางอย่างไม่ได้ หรือเพิ่มแล้วขัดแย้งกับข้อมูลเดิม

2.11.2.2 Second Normal Form (2NF)

ต้องเป็น First Normal Form (1NF) และต้องมี key (บางตำรา อาจจะเรียกว่า index) ที่ทุก Non-key จะต้องขึ้นอยู่ (depends on) กับ key นี้ และมีเพียง key เดียวในหนึ่งตาราง ซึ่งเรียกว่า Primary Key การที่ทุกตาราง (table) ต้องมี key ก็เพราะเราต้องการให้แน่ใจว่าทุกข้อมูลใน record ต่าง ๆ สามารถค้นหาได้โดยใช้ key

สรุปก็คือ นอร์มัลไลเซชันระดับที่ 2 (Second normal form : 2NF) เป็นการขจัดแอตริบิวที่ไม่ขึ้นกับทั้งส่วนของคีย์หลักออกไป เพื่อให้แอตตริบิวอื่นทั้งตรงกับส่วนที่เป็นคีย์หลักทั้งหมดเท่านั้น

นิยาม

เป็น First Normal Form (1NF) และทุก Non-key จะต้องขึ้นอยู่ (depends on) กับ key อย่างสมบูรณ์ (Full FD) หรืออาจกล่าวได้ว่าไม่มี Non-key ที่สามารถ imply ถึง Non-key ตัวอื่นได้ เช่น A -> (B, C) and B -> C รวมไปถึง การที่ Non-key บางตัวที่ขึ้นกับ บางส่วนของ key

ตัวอย่างเช่น

ABC (ชิ้นส่วน, ชื่อโกดัง, จำนวน, ที่อยู่โกดัง)

FD = {ชิ้นส่วน และ ชื่อโกดัง ->จำนวน, ชื่อโกดัง ->ที่อยู่โกดัง}

เนื่องจาก (ชิ้นส่วน และชื่อโกดัง) เป็น key แต่ (ที่อยู่โกดัง) ไม่ได้ขึ้นตรงกับ key (fully depended on key) ดังนั้น จึงไม่ใช้ 2NF ดังนั้น จะต้องทำการแตกรีเลชั่น เพื่อลดปัญหาความซ้ำซ้อนของข้อมูลเป็นดังนี้

สินค้า (ชิ้นส่วน, ชื่อโกดัง, จำนวน) โดยให้ ชิ้นส่วนและชื่อโกดัง เป็นคีย์หลัก

โกดัง (ชื่อโกดัง, ที่อยู่โกดัง) โดยให้ชื่อโกดัง เป็นคีย์หลัก

2.11.2.3 Third Normal Form (3NF)

นอร์มัลไลเซชั่นระดับที่ 3 (third normal form : 3 NF) คือ ขบวนการที่พยามขจัดสภาพของ Transitive Dependency ออกไป

นิยาม ของ นอร์มัลไลเซชั่นระดับที่ 3 (Third normal form : 3NF)

ต้องเป็น Second Normal Form (2NF) และไม่มี Transitive dependence หรือ เป็นการขจัดแอตตริบิวที่ไม่เป็นคีย์ที่ขึ้น (Transitive dependent) ตรงกับแอตตริบิวอื่นที่ไม่ใช่คีย์หลักออกไปเพื่อให้แอตตริบิวที่ไม่ใช่คีย์หลักต้องขึ้นตรงกับทั้งส่วนที่เป็นคีย์หลัก และไม่ขึ้นกับแอตตริบิวอื่นที่ไม่ใช่คีย์หลัก

นิยาม ของ Transitive dependency

การไม่ขึ้นตรงกับคีย์หลัก (Transitively Dependency) ถ้าในความสัมพันธ์ R มีคีย์หลักคือ K และแอตตริบิว A และ B จะกล่าวว่าแอตตริบิว B ไม่ขึ้นตรงกับคีย์หลัก

เมื่อ K -> A และ A -> B และ A -/-> K

ตัวอย่างการทำตารางให้เป็น 3NF

ผู้บริหาร (เลขประจำตัว, ชื่อนามสกุล, ที่อยู่, ตำแหน่ง, ยี่ห้อรถประจำตำแหน่ง)

FD = {เลขประจำตัว ->ชื่อนามสกุล, ที่อยู่, ตำแหน่ง

ตำแหน่ง -> ยี่ห้อรถประจำตำแหน่ง}

ในตัวอย่างจะเห็นได้ว่า set ของผู้บริหาร (เลขประจำตัว, ชื่อนามสกุล, ที่อยู่, ตำแหน่ง, ยี่ห้อรถประจำตำแหน่ง) นี้ ยังไม่ใช่ 3NF เพราะ เลขประจำตัว ->ตำแหน่ง ตำแหน่ง ->ยี่ห้อรถประจำตำแหน่ง ดังนั้น ควรจะแยก เซ็ตผู้บริหาร ออกเป็น 2 เซ็ต คือ

3NF : ผู้บริหาร (เลขประจำตัว, ชื่อนามสกุล, ที่อยู่, ตำแหน่ง)

ตำแหน่งบริหาร (ตำแหน่ง, ยี่ห้อรถประจำตำแหน่ง)

2.11.2.4 BCNF (Boyce/Codd Normal Form)

นิยาม

"ต้องเป็น 3NF และไม่มี attribute อื่นในรีเลชันที่สามารถระบุค่าของ attribute ที่เป็นคีย์หลัก หรือส่วนหนึ่งส่วนใดของคีย์หลักในการณีหลักเป็นคีย์ผสม"

โดยทั่วไปรูปแบบ BCNF จะอยู่ในรูปแบบ 3NF แต่ไม่จำเป็นเสมอไปที่รูปแบบ 3NF จะอยู่ในรูปแบบ BCNF ทั้งนี้เนื่องจากรูปแบบนี้เป็นการขยายขอบเขตของรูปแบบ 3NF ให้เหมาะสมยิ่งขึ้น โดยรูปแบบที่ต้องทำให้เป็น BCNF มักจะมีคุณสมบัติ ดังนี้

เป็น relation ที่มี key คู่แข่งหลายคีย์ (Multiple Candidate Key) โดยที่ คีย์คู่แข่งเป็นคีย์ผสม (Composite key) และ คีย์คู่แข่งนั้นมีบาวส่วนซ้ำซ้อนกัน (Overlapped) มี attribute บางตัวร่วมกันอยู่)

ตารางที่ 2.3 แสดง 3NF ที่ยังไม่เหมาะสม

รหัสนักศึกษา	ชื่อนักศึกษา	รหัสวิชา	เกรด
001	Jane	C01	A
001	Jane	C02	B
002	Timmy	C01	C
002	Timmy	C02	B
002	Timmy	C03	D
003	Nan	C04	D

จากตารางนี้จะได้ว่า

รหัสนักศึกษา, รหัสวิชา -> เกรด

ชื่อนักศึกษา, รหัสวิชา -> เกรด

รหัสนักศึกษา -> ชื่อนักศึกษา

ชื่อนักศึกษา -> รหัสนักศึกษา

ตารางนี้มี 4 determinants คือ (รหัสนักศึกษา, รหัสวิชา) (ชื่อนักศึกษา, รหัสวิชา) (รหัสนักศึกษา) และ (ชื่อนักศึกษา) แต่ตารางนี้มีเพียง 2 candidate keys คือ (รหัสนักศึกษา, รหัสวิชา) และ (ชื่อนักศึกษา, รหัสวิชา) สำหรับรหัสนักศึกษา และชื่อนักศึกษาไม่ใช่ Candidate key เราจะพบว่าตารางนี้เป็น 3NF ที่ไม่ดีพอเนื่องจากตารางนี้มี Candidate key 2 keys ด้วยกัน เราสามารถเลือกอันใดอันหนึ่งเป็น Primary key ได้ นอกจากนี้ Candidate key ทั้งสองยังเป็น composite key และ overlap กันอยู่เพราะมีรหัสวิชาร่วนกัน ทำให้มีข้อมูลหนึ่งชุดปรากฏหลายหนถึงแม้จะเป็น 3NF แล้วก็ตาม จึงควรปรับต่อไปเพื่อลดความซ้ำซ้อนนี้ลง และเพื่อแก้ปัญหาการ insert delete และ update ข้อมูลในตาราง โดยใช้วิธีการของ BCNF

1. ถ้ามี Transitive FD ให้ขจัด Transitive FD ทิ้งไป
2. attribute ตัวใดที่เป็น determinant ทำให้เป็น candidate key

จากตารางข้างต้นเราสามารถแยกออกเป็นตารางใหม่ ได้ 2 ตารางดังนี้

ตาราง นักศึกษา (รหัสนักศึกษา, ชื่อนักศึกษา)

ตาราง เกรด (รหัสนักศึกษา, รหัสวิชา, เกรด) หรือ (ชื่อนักศึกษา, รหัสวิชา, เกรด)

2.11.2.5 4NF (Forth Normal Form)

นิยาม

“ต้องอยู่ในรูปแบบ BCNF และเป็นรีเลชันที่ไม่มีความสัมพันธ์ในการระบุค่าของ attribute แบบหลายค่า โดยที่ attrubute ที่ถูกระบุค่าเหล่านี้ไม่มีความสัมพันธ์กัน (Independently Multivalued Dependency)

ตารางที่ 2.4 แสดงตารางที่เป็น Independently Mulivalued Dependency

Person	Project	Part	QtyUsed	HrsSpent
John	P1	Nut	11	7
Emmy	P1	Bolt	7	17
John	P1	Bolt	7	7
Emmy	P1	Nut	11	17

John	P2	Bolt	7	32
Jim	P2	Screw	9	45
John	P2	Screw	9	32
Jim	P2	Bolt	7	45

สามารถเขียน FDs ได้ดังต่อไปนี้

Project -> -> Part, QtyUsed

Project, Person -> HrsSpent

Project, Part -> QtyUsed

จะเห็นได้ว่าตารางนี้มีขึ้นต่อกันแบบหลายค่า (Multivalude Dependency) ซึ่งได้แก่ Part และ QtyUsed ที่ข้อมูลจะมีลักษณะเป็นคู่ ๆ ตาม Project ดังนั้น ตารางนี้จึงยังไม่อยู่ในรูปแบบ BCNF จะต้องทำการแตกตารางออกเป็น 2 ตารางออกเป็น 2 ตาราง ดังนี้

Person-Works-On-Project

ตารางที่ 2.5 แสดงตารางที่แยก Independently Mulivalued Dependency ออกแล้ว

Project	Person	HrsSpent
P1	John	7
P1	Emmy	17
P2	John	32
P2	Jim	45

FDs : Project, Person -> HrsSpent

Part-Used-In-Project

ตารางที่ 2.6แสดงตารางที่แยก Independently Mulivalued Dependency ออกแล้ว

Project	Part	QtyUsed
P1	Nut	11

P1	Bolt	7
P2	Bolt	7
P2	Screw	9

FDs : Project, Part -> QtyUsed

2.11.2.6 5NF (Fifth Normal Form)

5NF หรือเรียกว่า Project-Join Normal Form (PJ/NF)

นิยาม

“ต้องอยู่ในรูปแบบ 4NF และไม่มี Symmetric Constraint กล่าวคือ หากมีการแตก relation ออกเป็น relation ย่อย (Projection) และเมื่อทำการเชื่อมโยงรีเลชันย่อยทั้งหมด (Joint) จะไม่ก่อให้เกิดข้อมูลใหม่ที่ไม่เหมือน relation เดิม (Spurious Tupes)

ในการแตก relation ออกมาจากรูปแบบ 4NF นั้น ถ้าทำการเชื่อมโยง relation ย่อยนั้นใหม่ หากไม่มีข้อมูลที่แตกต่างไปจาก relation เดิม ก็จะสามารถแตก relation นั้นได้ แต่ถ้าหากเป็น relation ย่อยแล้วเกิดข้อมูลไม่เหมือนกับ relation เดิม ก็ไม่ควรแตก relation และให้ถือว่า relation เดิมอยู่ใน 5NF แล้ว

2.11.3 ประเด็นที่ควรคำนึงถึงในการทำให้อยู่ในรูปแบบ Normal Form

2.11.3.1 การแตกรีเลชันมากเกิดไป (Overnormolization)

วัตถุประสงค์ในการทำให้เป็นรูปแบบนอร์มัลไลซ์ก็คือ เพื่อลดปัญหาความซ้ำซ้อนของข้อมูลและลดปัญหาในเรื่องการเพิ่ม การปรับปรุง หรือลบข้อมูล โดยทั่วไปแล้วการออกแบบในระดับแนวความคิดผู้ออกแบบจะพยามวิเคราะห์ให้อยู่ในรูปแบบ 3NF แต่ถ้ามีกรณีของปัญหาที่จำเป็นต้องทำต่อไปถึงรูปแบบของ BCNF หรือ 4NF หรือ 5NF (ซึ่งเกิดขึ้นน้อยมากในทางปฏิบัติ) แต่อย่าพยามแตกรีเลชันให้มากเกินความจำเป็น เพราะการแตกรีเลชันออกมากเกินไปนั้นจะมีผลต่อประสิทธิภาพในการทำงานของฐานข้อมูลนั้น ๆ เช่น ในการค้าหาข้อมูลจะถูกใช้เวลามากเป็นต้น

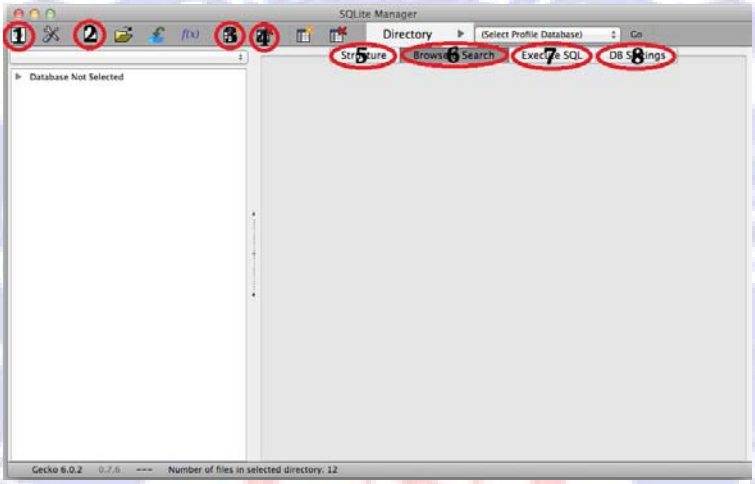
2.11.3.2 การดีนอร์มัลไลเซชัน (Denoralization)

ในกรณีที่บางรีเรชั่นถูกออกแบบโดยการไม่ทำให้อยู่ในรูปแบบบรรทัดฐานที่เป็นไปตามกฎเกณฑ์ที่กำหนดไว้ เช่น ควรปรับให้อยู่ในรูปแบบ 3 NF แต่หยุดอยู่เพียงแค่ 2NF เป็นต้น ทั้งนี้ อาจเป็นเพราะ

เหตุผลในเรื่องของประสิทธิภาพในการเรียกดู หรือค้นหาข้อมูล และยอมให้เกิดความซ้ำซ้อนของข้อมูลได้

ด้วยเหตุที่การดีนอร์มัลไลเซชัน อาจทำให้เกิดปัญหา ความซ้ำซ้อนของข้อมูล จึงควรมีการระบุสาเหตุ และวิธีการในการปรับปรุงข้อมูลในโปรแกรมประยุกต์ใช้งาน เพื่อป้องกันไม่ให้เกิดปัญหาข้อมูลไม่ถูกต้อง อีกประเด็นที่ยอมให้มีการดีนอร์มัลไลซ์หรือไม่ ก็คือ ถ้าข้อมูลในรีเลชันนั้น ๆ ส่วนใหญ่ จะเป็นการเรียกดูข้อมูล (Select) มากว่าการเพิ่ม ปรับปรุง หรือลบข้อมูล ก็อาจจะดีนอร์มัลไลเซชัน ถ้าคิดว่าการออกแบบลักษณะนี้จะเพิ่มประสิทธิภาพในการทำงานของญานข้อมูล และไม่มีปัญหาด้านความไม่ถูกต้ององข้อมูลที่ซ้ำซ้อนกันได้

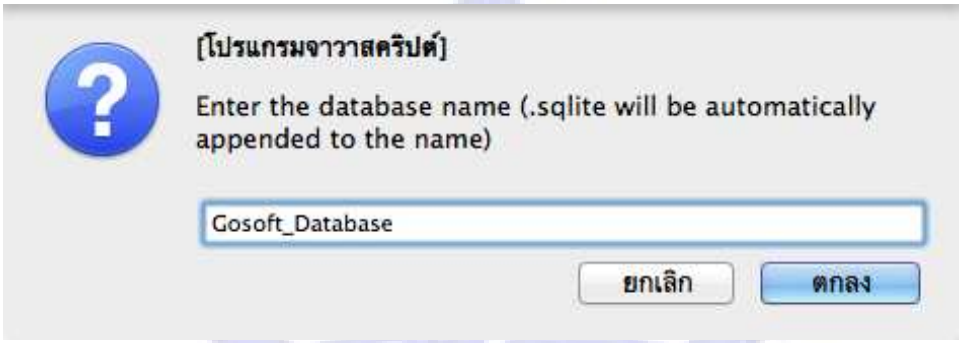
2.12 เครื่องมือ SQLite Manager



ภาพที่ 2.29แสดงหน้าแรกหลังเปิด SQLite Manager

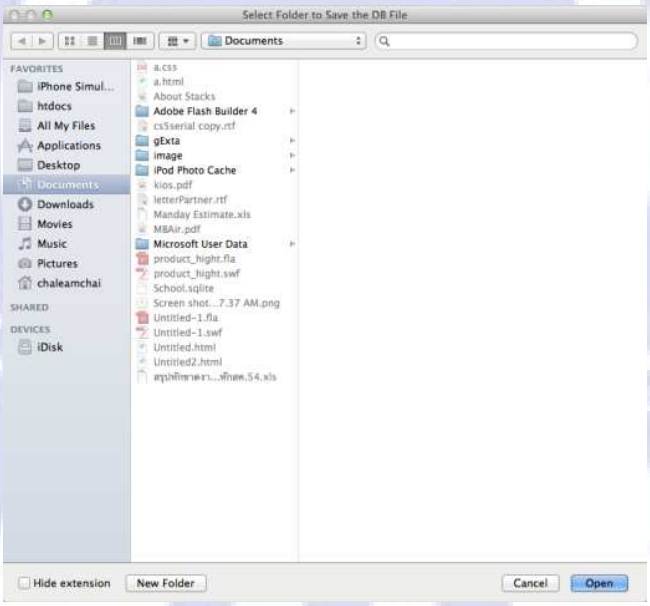
1. ปุ่มสำหรับการโหลดข้อมูลใหม่
2. สร้าง Database ตัวใหม่
3. สร้าง Table (ต้องมี database)อยู่ก่อน
4. ลบ Table
5. จัดการกับโครสร้างของ table ที่เลือกไว้

6. จัดการกับข้อมูลใน table
7. จัดการด้วยคำสั่ง SQL
8. จัดการ Database



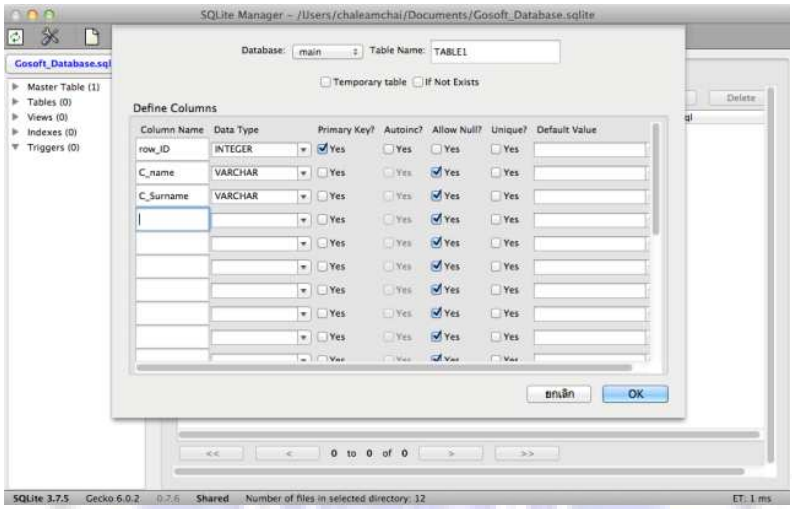
ภาพที่ 2.30 สร้าง Database ตัวใหม่

หลังจากกดปุ่ม เลข 2 จะเข้ามาสู่หน้าที่ให้กรอกชื่อ database ตัวใหม่



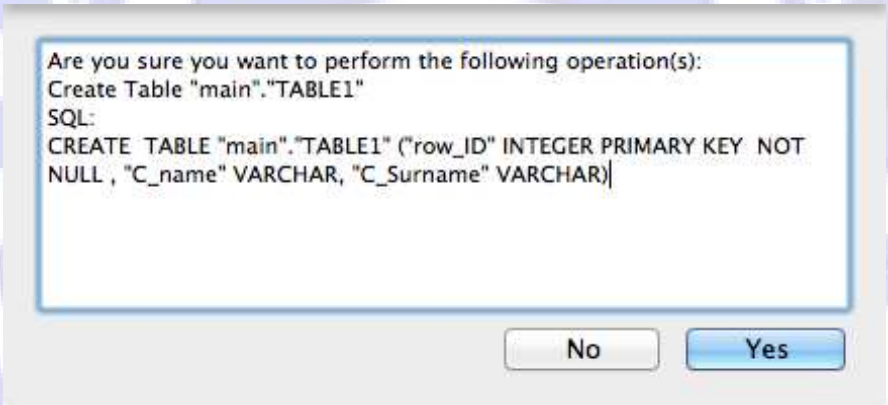
ภาพที่ 2.31 เลือกที่เก็บ Database ที่เราสร้างขึ้น

ให้เราเลือกสถานที่ ๆ จะเก็บ database ตัวใหม่ ในตัวอย่างเก็บที่ Documents



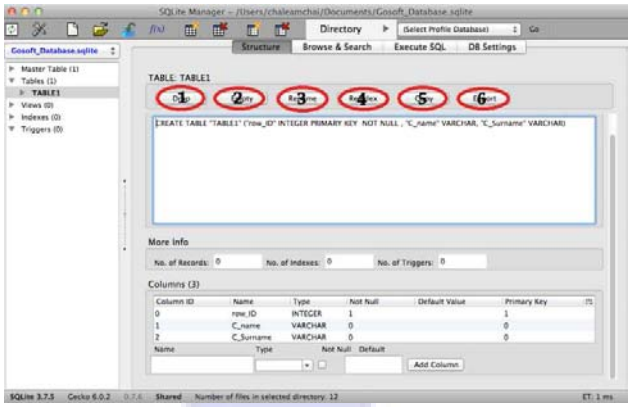
ภาพที่ 2.32 สร้าง Table

หลังจากกดปุ่มหมายเลย 3 จะเข้าสู่หน้าจะให้กรอกชื่อ, ประเภท ของ column



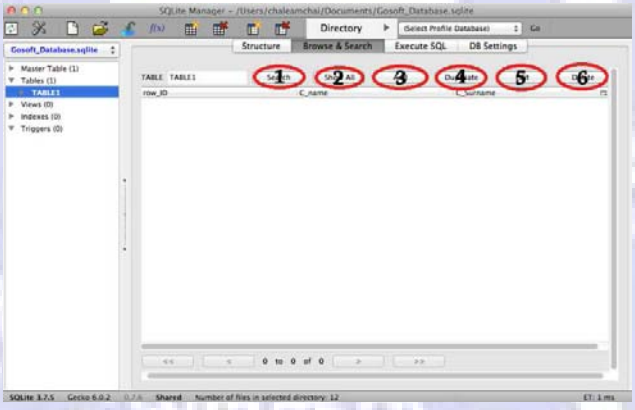
ภาพที่ 2.33 ยืนยันการสร้าง Table

ยืนยันว่าเราต้องการจะสร้าง table ตัวนี้



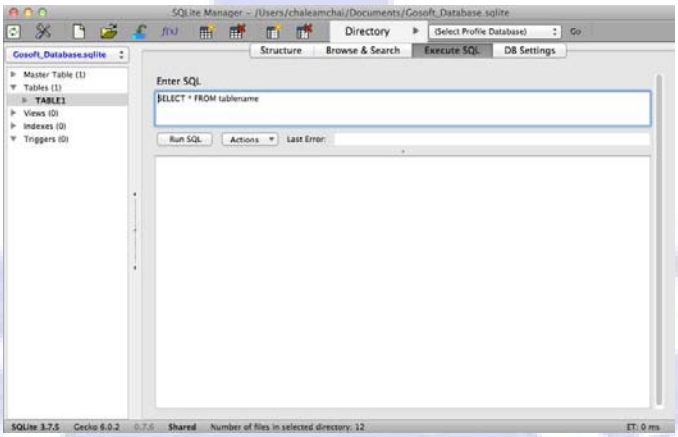
ภาพที่ 2.34จัดการกับโครงสร้าง Table

1. ลบ table ทิ้ง
2. ลบข้อมูลทั้งหมดใน table
3. แก้ไขชื่อ table
4. เรียงลำดับของแถวใหม่อีกครั้งโดยใช้ Primary Key
5. คัดลอก table
6. ส่งออก table



ภาพที่ 2.35จัดการกับข้อมูลใน Table

1. ค้นหาข้อมูลจากใน table เลือกไว้
2. แสดงข้อมูลทั้งหมดใน table
3. เพิ่มข้อมูลใหม่ลงใน table
4. ลอกเลียนแบบข้อมูล
5. แก้ไขข้อมูลในแถวที่เลือกไว้
6. ลบแถวของข้อมูลที่เลือกไว้



ภาพที่ 2.36การจัดการโดยใช้คำสั่ง SQL

สามารถใช้คำสั่ง SQL จัดการกับข้อมูลได้จากที่นี้

บทที่ 3
แผนงานการปฏิบัติงานและขั้นตอนการดำเนินงาน

3.1 แผนงานปฏิบัติงาน

กิจกรรม	มิถุนายน				กรกฎาคม				สิงหาคม				กันยายน			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
เก็บข้อมูลลูกค้า																
สร้าง prototype ให้ลูกค้า																
จัดทำ proposal																
Design																
Develop																
Testing& Debug																
สรุปผลการดำเนินงาน																

3.2 รายละเอียดงาน

พัฒนาซอฟต์แวร์ตัวอย่าง ให้กับพนักงานขายสินค้า (เน้นไปที่ข้าวตราฉัตร)เพื่อให้พนักงานมีความน่าเชื่อถือจากการใช้เทคโนโลยีใหม่ มีการทำงานที่มีแบบแผนตรงกัน ลดขั้นตอนการทำงานที่ต้องกลับมาบริษัทเพื่อกรอกข้อมูลลงฐาน

3.3 ขั้นตอนการดำเนินงาน

3.3.1 เก็บความต้องการของลูกค้า

ทำการคุยกับลูกค้าเพื่อเก็บความต้องการหลัก ๆ และตกลงระยะเวลาในการสร้างซอฟต์แวร์ตัวอย่างโดยความต้องการของลูกค้ามีดังนี้

1. บอกตำแหน่งร้านค้าและ ตำแหน่งปัจจุบันของผู้ใช้บนแผนที่
2. เพิ่มร้านค้าใหม่ลงบนแผนที่ได้โดยการกดบนแผนที่
3. บอกเส้นทางการเดินทางจากตำแหน่งที่อยู่ปัจจุบันไปถึงร้านค้าที่กำหนด

4. แสดงรายละเอียดสินค้าได้ว่ามีคุณสมบัติต่าง ๆ
5. เปรียบเทียบราคาระหว่างร้านกับร้าน หรือ สินค้ากับสินค้า
6. แสดงโปรโมชั่นต่าง ๆ ของร้านค้า
7. สั่งสินค้า
8. ถ่ายรูป

3.3.2สร้างซอฟต์แวร์ตัวอย่าง

สร้างซอฟต์แวร์ตัวอย่างให้ลูกค้าดูเพื่อช่วยตัดสินใจ โดยมีการทำงานครบทุกความต้องการของลูกค้าแต่ไม่เจาะลึกทุกการทำงาน และมีการจำลองข้อมูลต่าง ๆจากนั้นให้ลูกค้าตรวจดูโครงสร้างของโปรแกรมประยุกต์ที่สร้างขึ้นว่าตรงกับความต้องการหรือไม่ ต้องการจะเพิ่มส่วนใดอีก เมื่อตกลงงานแล้ว จึงจัดทำเอกสารข้อตกลง ค่าใช้จ่ายและระยะเวลาส่งมอบงาน

3.3.3วางแผนการดำเนินงาน

แบ่งการทำงานออกเป็นส่วนต่างตามความต้องการของลูกค้า ได้เป็น 2 ส่วนใหญ่ ๆคือส่วนจัดการฐานข้อมูล และส่วนติดต่อกับผู้ใช้งาน ดังนี้

ส่วนฐานข้อมูล

1. ออกแบบฐานข้อมูล
2. สร้างฐานข้อมูลบน iPad โดยใช้ SQLite ในส่วน ซอฟต์แวร์ตัวอย่างได้มีการจำลองข้อมูลลงไปเพื่อให้ง่ายต่อการแสดงผล
3. สร้างส่วนติดต่อระหว่างฐานข้อมูลกับ โปรแกรมประยุกต์
4. สร้างส่วนติดต่อระหว่างฐานข้อมูลบน iPad และ ฐานข้อมูลกลาง

ส่วนติดต่อผู้กับผู้ใช้งาน

1. แผนที่
2. เพิ่มร้านค้า
3. แสดงเส้นทาง
4. สั่งสินค้า
5. โปรโมชั่น
6. เปรียบเทียบราคา

7. ถ่ายรูป

มีการประชุมความคืบหน้าของงาน และอธิบายปัญหาที่เกิดขึ้นทุกสัปดาห์

3.3.4เริ่มการสร้างซอฟต์

ในส่วนของการสร้างนั้นแบ่งการทำงานออกเป็น 3 ส่วนคือ

1. Design ออกแบบฐานข้อมูลจากสิ่งที่ลูกค้าต้องการ โดยประกอบ
   - แสดงแผนที่ร้าน และสามารถเพิ่มร้านเองได้
   - เก็บข้อมูลลูกค้าพร้อมทั้งเก็บรูปที่ถ่ายไว้ได้
   - แสดงรายการสินค้าและ ราคาแต่ละสาขาพร้อมทั้งเปรียบเทียบราคาได้
   - ถ่ายรูปแล้วเก็บลงเครื่องสามารถเรียกขึ้นมาแสดงผลได้

3.3.4.1 การทำ normalization

หมายเหตุ : เพื่อให้เข้าใจจึงได้จำลองข้อมูล และตัดบาง column ออกไปเพื่อให้มีขนาดเล็กลง

ตารางที่ 3.1 แสดงข้อมูลจำลองของฐานข้อมูลแบบย่อ

Store_ID	StoreName	Branch	ManagerName	ManagerPhone	Category	Brand	Type	Kind	Size	Price
S1	CP Freah	สีลม	Sirichai	080551xxxx	ข้าว	ฉัตร	ข้าวสวย	ฉัตรทอง	5 kg	10
									10 kg	20
			Nisatron	089472xxxx	ข้าว	ฉัตร	ข้าวสวย	ฉัตรทอง	5 kg	10
		สามย่าน	Nisatron	089472xxxx	ข้าว	ฉัตร	ข้าวสวย	ฉัตรทอง	5 kg	10
					เครื่องดื่ม	แฟนต้า	น้ำอัดลม	สีแดง	1 cm^3	10

แปลงให้อยู่ในรูปแบบ 1NF โดยทุก ๆfield ในแต่ละ reconrd จะเป็น single value

ตารางที่ 3.2 แสดงข้อมูลจำลองของฐานข้อมูลแบบย่อในรูปแบบ 1NF

Store_ID	StoreName	Branch	ManagerName	ManagerPhone	Category	Brand	Type	Kind	Size	Price
S1	CP Freah	สีลม	Sirichai	080551xxxx	ข้าว	ฉัตร	ข้าวสวย	ฉัตรทอง	5 kg	10
S1	CP Freah	สีลม	Sirichai	080551xxxx	ข้าว	ฉัตร	ข้าวสวย	ฉัตรทอง	10 kg	20

S1	CP Freah	สีลม	Nisatron	089472xxxx	ข้าว	ฉัตร	ข้าวสวย	ฉัตรทอง	5 kg	10
S1	CP Freah	สามย่าน	Nisatron	089472xxxx	ข้าว	ฉัตร	ข้าวสวย	ฉัตรทอง	5 kg	10
S1	CP Freah	สามย่าน	Nisatron	089472xxxx	เครื่องดื่ม	แฟนต้า	น้ำอัดลม	สีแดง	1 cm^3	10

แปลงให้อยู่ในรูปแบบ 2NF โดยทุก ๆNon-Key ต้องมี Key อยู่และต้องมี key เดียวในตาราง

จากตาราง 1NF (Store_ID -> StoreName)

(StoreName -> Branch)

(Branch -> ManagerName, ManagerPhone)

(ManagerName -> Branch)

(ManagerName -> ManagerPhone)

(Category -> Brand)

(Brand -> Type)

(Type -> Kind)

(Kind -> Size)

(Size -> Price)

เพราะฉะนั้นจะได้ตารางใหม่เป็น

ตารางที่ 3.3แสดงตาราง Store ในรูป 2NF

store	
Store_ID	StoreName
S1	CPFresh

ตารางที่ 3.4แสดงตาราง Position ในรูป 2NF

Position	
Branch	Store_ID
สีลม	S1
สามย่าน	S1

ตารางที่ 3.5แสดงตาราง Manager ในรูป 2NF

Manager		
ManagerName	ManagerPhone	Branch
Sirichai	080551xxxx	สีลม
Nisatron	089472xxxx	สามย่าน

ตารางที่ 3.6แสดงตาราง Category ในรูป 2NF

Category
Category
ข้าว
เครื่องดื่ม

ตารางที่ 3.7แสดงตาราง Brand ในรูป 2NF

Brand	
Brand	Category
ฉัตร	ข้าว
แฟนต้า	เครื่องดื่ม

ตารางที่ 3.8แสดงตาราง Type ในรูป 2NF

Type	
Type	Brand
ข้าวสวย	ฉัตร
น้ำอัดลม	แฟนต้า

ตารางที่3.9แสดงตาราง Kind ในรูป 2NF

Kind	
Kind	Type
ฉัตรทอง	ข้าวสวย

สีแดง	น้ำอัดลม

ตารางที่ 3.10แสดงตาราง Size ในรูป 2NF

Size	
Size	Kind
5kg	ฉัตรทอง
10kg	ฉัตรทอง
1 cm^3	น้ำอัดลม

ตารางที่ 3.11แสดงตาราง Price ในรูป 2NF

Price	
Price	size
10	5kg
20	10kg
10	1 cm^3

ทำการแปลงให้อยู่ในรูป 2NF แล้วจะเห็นว่าCandidate key แล้วจึงถือว่าเป็น 3NF แล้ว

3.3.4.2 การเชื่อมต่อฐานข้อมูลส่วนกลาง

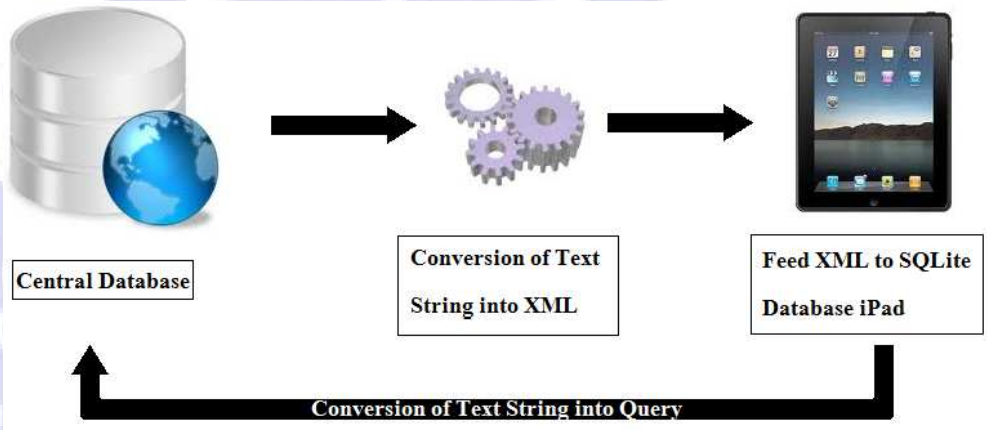
การเชื่อมต่อฐานข้อมูลนั้นทำได้ 2 กรณีคือ

1. เมื่อต้องการข้อมูลใหม่จากฐานข้อมูลกลาง feed มาที่ iPad

   เมื่อมีการเปิดโปรแกรมจะทำการตรวจว่าเครื่องมีการ Connect Internet หรือไม่ถ้ามีจะทำการเชื่อมต่อไปที่ฐานข้อมูลส่วนกลางเพื่อขอข้อมูลใหม่ โดยเครื่องจะส่งเวลาที่ได้จากตาราง LastConnectToServer ไป และฐานข้อมูลส่วนกลางจะ SELECT ข้อมูลที่ใหม่กว่าเวลานั้นออกมาส่งกลับมาในรูปแบบ XML จากนั้นจะทำการ feed ข้อมูลและเก็บลงใน Database พร้อมบันทึกเวลาที่ทำงานเสร็จสิ้นลงในตาราง LastConnectToServer

2. เมื่อต้องการส่งข้อมูลลูกค้าที่เพิ่งเก็บลง iPad สู่ฐานข้อมูล

ในการส่งข้อมูลลูกค้าใหม่ขึ้นไปนั้นเครื่องจะทำการ SELECT ข้อมูลที่ใหม่กว่าเวลาที่ถูกเก็บไว้ในตาราง LastConnectToServer มาใช้ ต่อมาข้อมูลจะถูกแปลงให้อยู่ในรูปคำสั่ง SQL เพื่อจะส่งไปได้ใน Link เดียว และจะรอการตอบกลับจากฐานข้อมูลส่วนกลาง โดยฐานข้อมูลจะทำการ SELECT ข้อมูลที่ใหม่กว่าเวลาที่ได้จาก Link (เวลาจากตาราง LastConnectToServer) และส่งกลับมาในรูปแบบ XML เมื่อได้รับจะทำการ feed ข้อมูลและเก็บลงใน Database โดยจะลบข้อมูลที่ส่งขึ้นไปจาก Databaseในเครื่องก่อนจากนั้นจะเก็บข้อมูลที่ได้มาใหม่ลงไปแทน พร้อมบันทึกเวลาที่ทำงานเสร็จสิ้นลงในตาราง LastConnectToServer



ภาพที่ 3.1แสดงการทำงานเมื่อส่งข้อมูลจาก Database สู่เครื่องและเครื่องสู่ Databas

2. Develop

เริ่มสร้าง Database ตามที่ได้ออกแบบไว้เบื้องต้นด้วย SQLite Manager จากนั้นก็นำไปรวมเข้ากับโปรแกรม Xcode เขียนส่วนเชื่อมต่อระหว่าง Database กับ application โดยคิดจากการทำงานคือ เมื่อเปิดโปรแกรมขึ้นมาต้องมีการแสดงร้านค้าบนแผนที่จึงมีการ query ตำแหน่งที่ตั้งของร้านออกมาใช้เป็นต้น

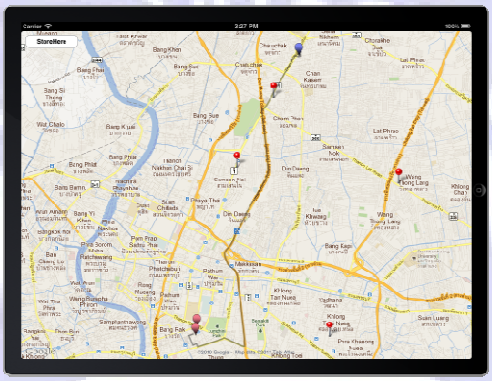
3. Testing & Debug

หลังจากสร้างเสร็จแล้วได้ให้ที่ปรึกษาโตรงการ ดูโครงสร้างที่ออกแบบไว้ ว่าเหมาะสมและเพียงพอต่อความต้องการของลูกค้าหรือไม่เมื่อได้รับคำชี้แนะ จะนำกลับไปแก้ไขและตรวจสอบใหม่ตามลำดับ

บทที่ 4
ผลการดำเนินงาน การวิเคราะห์และสรุปผลต่าง ๆ

4.1 ขั้นตอนและการดำเนินงาน

เมื่อเริ่มโปรแกรมจะแสดงหน้าแผนที่ขึ้นมาโดยนำข้อมูลจากตำแหน่งร้านค้าต่าง ๆ จากฐานข้อมูลมาแสดงผลและมีการตรวจสอบฐานข้อมูลกลางว่ามีข้อมูลใหม่หรือไม่ ถ้ามีจะทำการสำเนามา โดยวิธีการสำเนานั้นจะส่งเวลาครั้งสุดท้ายที่เครื่องติดต่อฐานข้อมูลกลางไปจากตาราง LastConnectToServer จากนั้นฐานข้อมูลจะทำการเลือกข้อมูลที่ใหม่กว่าเวลานั้นส่งกลับมาให้

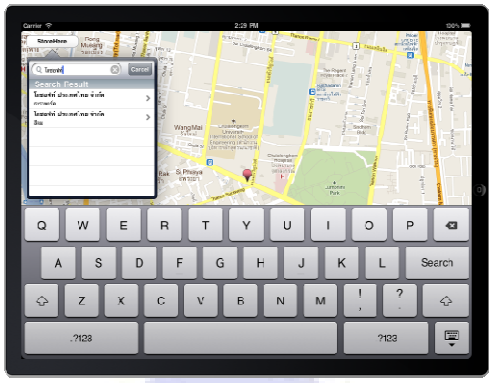


ภาพที่ 4.1 แสดงตำแหน่งร้านที่ดึงมาจากฐานข้อมูล

หน้าเมนูจะมี 3 เมนูให้เลือกคือ

1. ค้นหาร้านค้า (Search store)

เมื่อเลือกเข้าไปแล้วจะมีการดึงข้อมูลของร้านค้าต่าง ๆ มาแสดงผลประกอบไปด้วยที่ตั้งและข้อมูลของร้านค้า ภายใน cell จะแสงแค่ชื่อร้านและ สาขา แต่ว่า Search นั้นสามารถค้นหาจากคีย์ที่ดึงออกมาได้ (ที่ตั้ง และข้อมูลร้านค้า)



ภาพที่ 4.2 แสดงเมนูค้นหาร้านค้า

เมื่อเลือกเข้าไปใน Cell ใดก็ตามแผนที่จะทำการเลื่อนมุมมองไปยังร้านที่เลือก พร้อมกับหน้าเมนูค้นหาร้านค้าเปลี่ยนสู่ขั้นตอนที่สอง เรียกว่าหน้า Detail (รายละเอียดของลูกค้า)

ภาพที่ "4.3" แสดงหน้ารายละเอียดลูกค้าพร้อมทั้งบอกตำแหน่งปุ่มเรียกฟังค์ชัน

โดยในหน้านี้จะนำข้อมูลที่ได้ query จากหน้าที่แล้วมาแสดงผลเฉพาะร้านที่ถูกเลือกในหน้าจะมีส่วนที่ส่วนการทำงานที่สำคัญ 3 อย่างคือ เลือกรูป, ขอเส้นทาง และสั่งสินค้า

1.1 เลือกรูป

การเลือกรูปนั้นมาจาก 2 ทาง คือจากการถ่ายรูปหรือทำการเลือกจากอัลบั้มรูปภายในเครื่องมาใช้

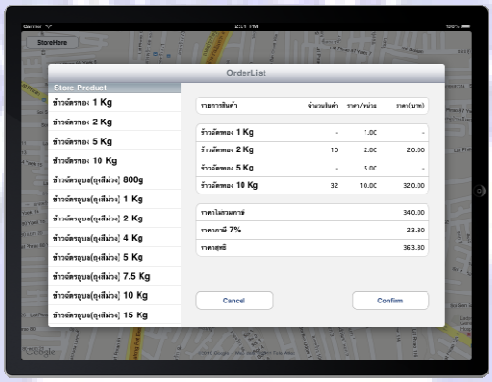
การถ่ายรูปนั้นจะมีการตรวจสอบด้วยว่าเครื่องมีกล้องหรือไม่ถ้าไม่มีกล้องจะฟ้องว่าใช้ไม่ได้หลังจากถ่ายรูปแล้วจะมีการแปลงข้อมูลของรูปภาพเป็น NSData แล้วเก็บลงในฐานข้อมูล

1.2 ขอเส้นทาง

เมื่อกดขอเส้นทางจะมีการส่งตำแหน่งของเราและจุดที่เลือกขึ้นไปที่ google เพื่อขอเส้นทางจากนั้นนำข้อมูลที่ได้มาแปลงให้เป็นเส้นทางแสดงบนแผนที่

1.3 สั่งสินค้า

ในหน้าสั่งสินค้าจะมีการ Query ออกมาแสดงผลในตารางด้ายซ้ายดัง**ภาพที่ 4.4** โดยสินค้าที่แสดงนั้นเป็นสินค้าของ iStore เท่านั้นและเมื่อเลือกขึ้นมาจะนำเข้าสู่ตารางด้านขวาที่สามารถ กรอกจำนวนชิ้นของสินค้าทั้งยังคิดราคารวมของสินค้าแต่ละขึ้นและราคารวมทั้งหมดด้วย

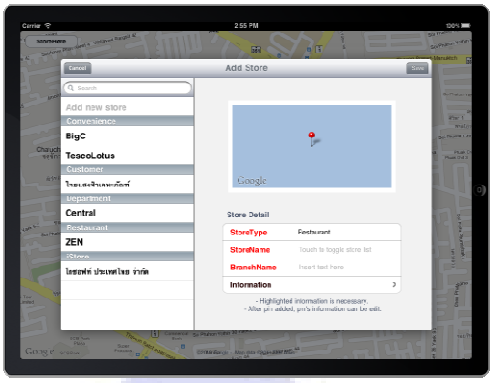


ภาพที่ 4.4 แสดงหน้าสั่งสินค้า

2. เพิ่มร้านค้า

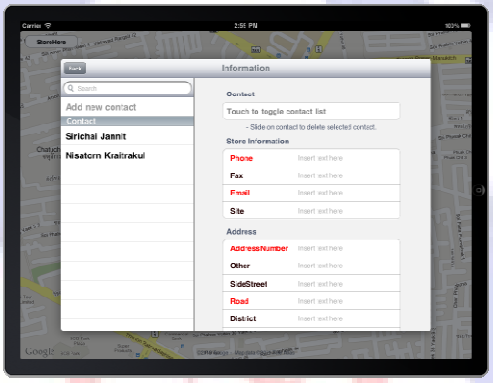
เมื่อเลือกเข้ามาจะมีช่องให้กรอกข้อมูลโดยแบ่งออกเป็น 2 ส่วนคือหน้าหลัก และหน้ารอง

หน้าหลักนั้นจำเป็นที่จะต้องกรอกข้อมูลครบถ้วนเพื่อจะบันถึกตำแหน่งร้านลงบนแผนที่ โดยช่อง Store name เมื่อกดเข้าไปจะมีตัวเลือกให้คือร้านที่มีอยู่ในฐานข้อมูลอยู่แล้วหรืออยากจะเพิ่มร้านใหม่เข้าไปก็ได้ หากเพิ่มร้านใหม่จะต้องเลือก Store type ด้วย



ภาพที่ 4.5 แสดงหน้าเพิ่มร้านค้า หน้าหลัก

หน้ารองนั้น จะใส่ หรือไม่ก็ได้เพียงแต่ถ้าไม่ใส่ หรือใส่ไม่ครบตามหัวข้อที่กำหนดร้านค้าใหม่ที่เพิ่มเข้าไปนั้นจะอยู่ในสถานะไม่ใช่ลูกค้า และไม่สามารถสั่งสินค้าได้ในช่องของ Contact นั้นถ้าไม่ใช่ร้านใหม่ จะมีการแสดงบุคคลที่ดูแลร้าน ในสาขาอื่น ๆ ให้เลือกและสามารถเพิ่มเข้าไปใหม่ได้เช่นกัน

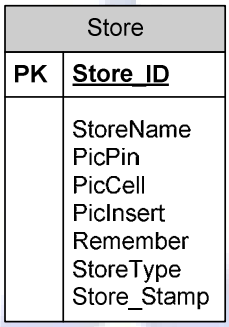


ภาพที่ 4.6 แสดงหน้าเพิ่มสินค้า หน้ารอง

เมื่อเลือกครับแล้วจะสามารถบันทึกลงสู่ฐานข้อมูลได้ในหน้าหลักกด save จะมีการถามเป็นครั้งสุดท้ายก่อนบันทึกลงเพราะเมื่อบันทึกไปแล้วจะไม่สามารถแก้ไขได้อีก

ในการบันทึกลงฐานข้อมูลนั้นจะมีการบันทึกทั้งหมด 4 ตารางประกอบไปด้วย Store, StorePosition, Contact, Managerและในการใส่ แต่ละตารางนั้นมีเงื่อนไขไม่เหมือนกัน

ตาราง Store ประกอบไปด้วยข้อมูลดังภาพที่ 4.7ซึ่งเงื่อนไขในการเก็บลงคือ ชื่อร้านใหม่จะต้องไม่ซ้ำกับชื่อร้านเดิมที่อยู่ในฐานข้อมูล (หรือก็คือร้านที่เลือกมาจากรายการ)



ภาพที่ 4.7แสดงข้อมูลภายใน ตาราง Store

ตาราง StorePosition ประกอบไปด้วยข้อมูลดังภาพที่ 4.8 โดยจะเก็บข้อมูลที่เก็บนั้นมาจากการกรอก และมี forgien key จากตาราง Store อยู่เพราะฉะนั้นจะต้องนำ Store_ID มาเก็บโดยเปรียบเทียบชื่อร้านว่าตรงกับStoreName แถวใด

ภาพที่ 4.8แสดงข้อมูลภายในตาราง StorePosition

ตาราง Contact และ Manager ตารางนี้ต้องอ้างอิงก่อนว่าทีมพัฒนาได้มากว่าเป็บความสัมพันธ์แบบ M : M ซึ่งก็คือสาขาหนึ่งมีคนดูแลได้หลายคน และหนึ่งคน ดูแลได้หลายสาขาจึงต้องมีตาราง Manager

มารองรับความสัมพันธ์นี้ การเก็บตาราง Contact นั้นจะเก็บลงนั้นจะมีการเก็บหลายครั้งตามที่ผู้ใช้ได้เพิ่ม Contact เข้ามา แต่จะไม่เก็บซ้ำกับชื่อที่มีอยู่แล้วในร้านนั้น กล่าวคือ Contact ที่เพิ่มโดยเลือกจากรายการนั้นจะไม่ทำการเพิ่มข้อมูลในฐานข้อมูล จะเพิ่มเฉพาะ รายชื่อใหม่เท่านั้น หลังจากเพิ่มตาราง Contact เสร็จแล้วจะทำการเพิ่มตาราง Manager

ContactName	
PK	**Contact_ID**
	ContactName ContactSurname Phone Email Contact_Stamp

ภาพที่ 4.9แสดงข้อมูลภายในตาราง Contact

ตาราง Manager นั้นประกอบไปด้วย forgien key 2 ตัวจากตาราง store และตาราง contact วิธีการคือนำ Store_ID ของตาราง Store ที่ StoreName ตรงกับร้านที่เพิ่มเข้าไป และ Contact_ID จากตาราง Contact ที่ ContactName ตรงกับที่เพิ่มเข้าไป เพื่อสร้างความสัมพันธ์แบบ M : M

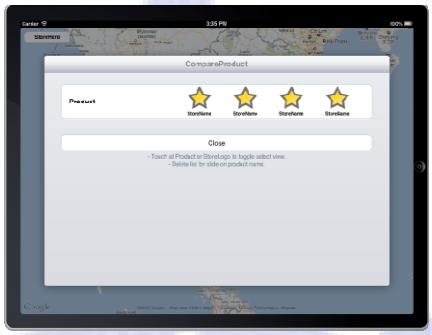
Manager	
PK	**Manager_ID**
FK1 FK2	Position_ID Contact_ID Manager_Stamp

ภาพที่ 4.10แสดงข้อมูลภาพในตาราง Manager

หลังจากเพิ่มเสร็จแล้วจะเข้าสู่อีกเงื่อนไขคือถ้ามี Internet จะมีการส่งข้อมูลขึ้นไปยังฐานข้อมูลกลางพร้อมกับเวลาเชื่อมต่อครั้งสุดท้าย จากตาราง LastConnectToServer ฐานข้อมูลกลางจะทำการบรรจุข้อมูลใหม่ลงไปและ เลือกข้อมูลที่ใหม่กว่าเวลาที่ส่งมา กลับคืนไปยังเครื่องของผู้ใช้

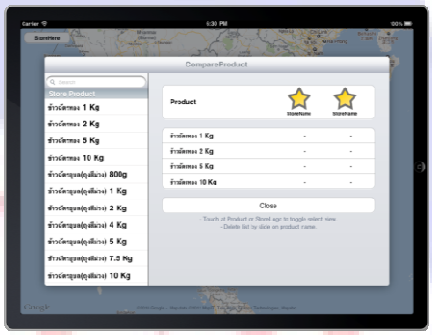
3. เปรียบเทียบราคาสินค้า

เมื่อเลือกเข้าจะมีการดึงข้อมูลสินค้า และร้านค้าทั้งหมดที่ถูกเก็บอยู่ในฐานข้อมูลออกมาแสดงผลเรียงตามตัวอักษร และใช้งานในแต่ละส่วน



ภาพที่ 4.11 แสดงหน้าเปรียบเทียบสินค้า

ส่วนที่ 1 คือเลือกสินค้ามาแสดงผลเมื่อเลือกสินค้าแล้วจะถูกนำไปแสดงผลในรายการด้านขวา



ภาพที่ 4.12 แสดงการเลือกสินค้า

ส่วนที่ 2 คือเลือกร้านค้ามาแสดงผลเมื่อเลือกร้านค้าแล้วจะถูกนำไปแสดงผลในรายการด้านขวา

ภาพที่ 4.13 แสดงการเลือกร้านค้า

เมื่อเลือกครบทั้ง 2 รายการแล้วจะเห็นตัวเลขของราคาสินค้าขึ้นมาแสดงผลโดยจะให้แถวแรกที่แสดงนั้นเป็นแถวหลัก และแถวต่อ ๆ มาเป็นแถวรอง แสดงได้สูงสุด 4 ร้านสามารถเลือกสินค้าได้เท่าที่ต้องการ จากภาพที่ 4.13 จะเห็นได้ว่าแถวรองมีการเปลี่ยนสีของราคาสินค้าเมื่อมีราคาถูกกว่า

4.2 ผลการวิเคราะห์ข้อมูล

ปัจจุบันพนักงานขายนั้นต้อง พกเอกสารสำหรับการสมัคสมาชิกที่ต้องเขียนลงในกระดาษ ด้วยปากกา และรายละเอียดสินค้ามากมายที่ใช้ประกอบการขาย ทำให้ดูไม่เหมาะสมกับภาพลักษณ์ที่เห็นผู้นำทางนวัตกรรม หลังจากการสำรวจลูกค้าเสร็จแล้ว พนักงานขายต้องกลับมาที่บริษัทเพื่อกรอกข้อมูลพวกนั้นเก็บลงสู่ฐานข้อมูลกลาง ซึ่งอาจจะมีข้อผิดพลาดเกิดขึ้นได้

จากการสร้างโปรแกรมประยุกค์นี้ทำให้ พนักขาย มีภาพลักษ์ดีขึ้นตรงวัตถุประสงค์การเป็นผู้นำขององค์กรทางด้านนวัตกรรมและ พนักงานสามารถทำงานได้ง่ายมากขึ้นไม่มีการทำงานซ้ำซ้อนที่จะต้องกลับมากรอกข้อมูลที่บริษัทอีกครั้งเพราะมีการส่งข้อมูลออนไลน์ของโปรแกรมประยุกต์ กับ ฐานข้อมูล

4.3 วิเคราะห์และวิจารณ์ข้อมูลโดยเปรียบเทียบผลที่ได้รับกับวัตถุประสงค์และจุดมุ่งหมายในการปฏิบัติงานหรือการจัดทำโครงการ

โปรแกรมประยุกค์นี้ยังไม่สมบูรณ์และมีปัญหาจากการที่ผู้ใช้ไม่รู้ถึงความสามารถทั้งหมดของโปรแกรมปรุยุกค์นี้ถ้าไม่มีคนอธิบายการใช้ก่อน อีกทั้ง มีปัญหากับการติดต่อฐานข้อมูลส่วนกลางเมื่อไม่ได้เชื่อมต่อกับอินเทอร์เน็ต

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

จากการดำเนินงานพบปัญหามากมายในการสร้างโปรแกรมประยุกต์นี้เนื่องจากผู้พัฒนาไม่เคยศึกษาเครื่องมือในการช่วยสร้างโปรแกรมคือXCode และภาษาในการพัฒนาคือ Objective-C มาก่อนจึงทำให้การทำงานเป็นไปอย่างเชื่องช้า ในตอนต้นหลังจากที่ได้รับความต้องการของลูกค้านั้นทีพัฒนาได้มีการออบแบบส่วนแสดงผลโดยการเขียนในการดาษแต่ด้วยความรู้ไม่เพียงพอจึงทำให้ส่วนการแสดงผลที่ออกแบบไว้ตอนต้นนั้นไม่สามารถทำให้ตรงตามความต้องการได้ ในทีมพัฒนาโปรแกรมเองนนั้นไม่ได้มีการแบ่งการทำงานอย่างชัดเจนจึงทำให้เกิดการทำงานซ้ำซ้อนอีกทั้งความต้องการของลูกค้าที่เปลี่ยนแปลงบ่อยครั้ง

5.2 แนวทางการแก้ไขปัญหา

ทางทีมผู้พัฒนาได้ศึกษาข้อมูลของเครื่องมือและ ภาษาผ่านทางสื่อการเรียนรู้ออนไลน์ และหนังสือต่าง ๆ การออกแบบส่วนการแสดงผลนั้นได้ออกแบบใหม่จากความรู้ที่มีอยู่แต่ยังยึดตามหลักทำให้เหนือกว่าความคาดหวังของลูกค้า จากนั้นเริ่มมีการแบ่งการทำงานเป็นส่วนต่าง ๆ ให้ชัดเจน และมีการตกลงกับลูกค้าเรื่องความต้องการเป็นเอกสาร

5.3 ข้อเสนอแนะจากการดำเนินงาน

จากการทำงานนั้นทำให้ทีมงานได้รู้ว่า application ที่เขียนนั้นสามารถทำงานได้เฉพาะบนเครื่อง iOS เท่านั้นทำให้การทำงานไม่หลากหลายหากต้องการใช้บน Smart phone อื่น ๆ อย่าง Android จึงควรจะนำ application นี้ไปประยุกต์ใช้บน Android ด้วย

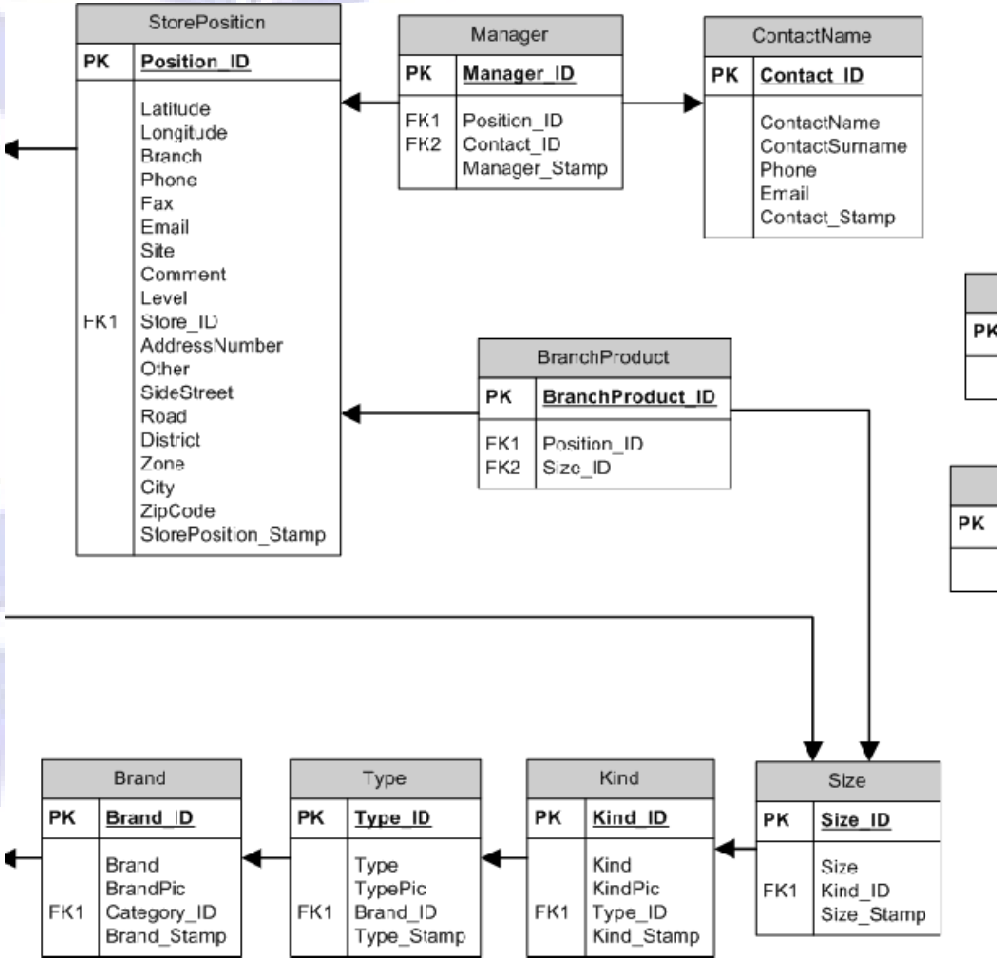
ในส่วนของความสามารถนั้น อยากให้เพิ่มความสามารถเปรียบเทียบราคาของผลิตภันฑ์ประเภทเดียวกัน ในร้านประเภทเดียวกันได้ เช่น ข้าวมีหลายยี่ห้อ ซึ่งแต่ละยี่ห้อนั้นราคาไม่เท่ากัน เพื่อให้ง่ายต่อการนำเสนอ และตัดสินใจของลูกค้า ส่วนการถ่ายรูปนั้นเก็บได้เพียงแค่ 1 รูปเท่านั้นอาจจะน้อยเกินไปในการเก็บภาพของร้านที่มีสาขาหลายสาขาจึงควรจะเก็บรูปได้มากกว่า 1 รูปใน 1 สาขา

เอกสารอ้างอิง

1. SQL Tutorial [Online], Avariable : http://www.thaicreate.com/tutorial/sql.html [2011, September 12].
2. Objective-C [Online], Avariable : http://en.wikipedia.org/wiki/Objective-C [2011, September 12].
3. Xcode 4 [Online], Avariable : http://www.idevthai.com/xcode4/ [2011, September 12].
4. Stackoverflow [Online], Avariable http://www.stackoverflow.com [2011, September 12].
5. iPhone Dev SDK [Online], Avariable http://www.iphonedevsdk.com [2011, September 12].
6. iPhone SDK Article [Online], Avariable http://www.iphonesdkarticles.com [2011, September 12].
7. Khomkrit Article [Online], Avariable http://khomkrit.blogspot.com [2011, September 12].
8. Xcode Tutorials [Online], Avariable http://www.xcode-tutorials.com/parsing-xml-files[2011, September 12].
9. Apple Developer Forums [Online], Avariable https://discussions.apple.com/community/developer_forums[2011, September 12].
10. Youtuble [Online], Avariable http:// www.youtube.com [2011, September 12].
11. Mark Dalrymple and Scott Knaster, 2009, Learn Objective-C on the Mac, พิมพ์ครั้งที่ 1, สำนักพิมพ์ Apress.
12. Dave Mark and Jeff Lamarche, 2009 , Beginning iPhone Development Exploring the iPhone SDK, พิมพ์ครั้งที่ 1, สำนักพิมพ์ Apress
13. Maher Ali, 2009, iPhone SDK Programming, พิมพ์ครั้งที่ 1, สำนักพิมพ์ WHLEY
14. Erica Sadun, 2009, The iPhone Developer's Cookbook, พิมพ์ครั้งที่ 1, สำนักพิมพ์ Addison-Wesley

15. ชาญชัย ศุภอรรถกร, 2553, สร้างเว็บอีคอมเมิร์ชด้วย PHP+MySQL ฉบับสมบูรณ์,สำนักพิมพ์ ธิงค์ บียอนด์ บุ๊คส์, บจก.
16. ศุภชัย สมพานิชม, 2553, Basic ASP.NET 4.0,สำนักพิมพ์ ซิมพลิฟาย
17. รวิทัต ภู่หลำ, 2554,คู่มือเขียน iPhone Apps,สำนักพิมพ์ Provision

ภาคผนวก


StorePosition
PK Position_ID
Latitude
Longitude
Branch
Phone
Fax
Email
Site
Comment
Level
FK1 Store_ID
AddressNumber
Other
SideStreet
Road
District
Zone
City
ZipCode
StorePosition_Stamp
Manager
PK Manager_ID
FK1 Position_ID
FK2 Contact_ID
Manager_Stamp
ContactName
PK Contact_ID
ContactName
ContactSurname
Phone
Email
Contact_Stamp
BranchProduct
PK BranchProduct_ID
FK1 Position_ID
FK2 Size_ID
Brand
PK Brand_ID
Brand
BrandPic
FK1 Category_ID
Brand_Stamp
Type
PK Type_ID
Type
TypePic
FK1 Brand_ID
Type_Stamp
Kind
PK Kind_ID
Kind
KindPic
FK1 Type_ID
Kind_Stamp
Size
PK Size_ID
Size
FK1 Kind_ID
Size_Stamp

ประวัติผู้จัดทำ

ชื่อ-สกุล นายศิริชัย จันทร์นิตย์

วันเดือนปีเกิด 16 สิงหาคม 2532

ประวัติการศึกษา

ระดับประถมศึกษา	โรงเรียนราชวินิต	พ.ศ. 2544
ระดับมัธยมศึกษา	โรงเรียนเทพศิรินทร์	พ.ศ. 2550