แอพพลิเคชั่นการเงินส่วนบุคคล

PERSONAL FINANCE APPLICATION

นวพงศ์ อารีกุล

กันตพัฒน์ เลิศอำไพกุลวงศ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2558

แอพพลิเคชั่นการเงินส่วนบุคคล

PERSONAL FINANCE APPLICATION

นวพงศ์ อารีกุล

กันตพัฒน์ เลิศอำไพกุลวงศ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2558

ปริญญานิพนธ์ปีการศึกษา 2558
ภาควิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
เรื่อง แอพพลิเคชั่นการเงินส่วนบุคคล

PERSONAL FINANCE APPLICATION

ผู้จัดทำ
1. นายนวพงศ์ อารีกุล รหัสนักศึกษา 54010685
2. นายกันตพัฒน์ เลิศอำไพกุลวงศ์ รหัสนักศึกษา 55010071

อาจารย์ที่ปรึกษา
(รศ. เจริญ วงษ์ชุ่มเย็น)

อาจารย์ที่ปรึกษา
(ดร.ปกรณ์ วัฒนจตุรพร)

ปริญญานิพนธ์ปีการศึกษา 2558
ภาควิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
เรื่อง แอพพลิเคชั่นการเงินส่วนบุคคล

PERSONAL FINANCE APPLICATION

ผู้จัดทำ
1. นายนวพงศ์ อารีกุล รหัสนักศึกษา 54010685
2. นายกันตพัฒน์ เลิศอำไพกุลวงศ์ รหัสนักศึกษา 55010071

อาจารย์ที่ปรึกษา
(รศ. เจริญ วงษ์ชุ่มเย็น)

อาจารย์ที่ปรึกษา
(ดร.ปกรณ์ วัฒนจตุรพร)

แอพพลิเคชั่นการเงินส่วนบุคคล

นายนวพงศ์	อารีกุล	54010685
นายกันตพัฒน์	เลิศอำไพกุลวงศ์	55010071
รศ. เจริญ	วงษ์ชุ่มเย็น	อาจารย์ที่ปรึกษา
ดร. ปกรณ์	วัฒนจตุรพร	อาจารย์ที่ปรึกษาร่วม

ปีการศึกษา 2558

บทคัดย่อ

โครงงานนี้จัดทำเพื่อศึกษาออกแบบ และพัฒนาแอพพลิเคชั่นสำหรับการวางแผนการเงินส่วนบุคคลเพื่อให้บุคคลทั่วไปที่สนใจด้านการวางแผนการเงินส่วนบุคคลสามารถจัดทำงบประมาณส่วนบุคคลได้และเพื่อให้บรรลุเป้าหมายการเงินที่วางแผนเอาไว้ได้ แอพพลิเคชั่นนี้พัฒนาให้รองรับกับระบบ iOS สำหรับ iPhone ซึ่งในแอพพลิเคชั่นนั้นจะมีตั้งแต่การตั้งเป้าหมายการเงินแล้วคำนวณเป็นงบประมาณที่ต้องออมเพื่อบรรลุเป้าหมายนั้นรวมถึงการจัดทำงบการเงินส่วนบุคคลเพื่อที่รวบรวมข้อมูลทางการเงินไว้ในที่เดียว เพื่อที่จะสามารถวิเคราะห์ในด้านการเงินได้ เช่น สถานะการเงิน ที่มาของเงิน และการเงินของเราทั้งหมด

แอพพลิเคชั่นการเงินส่วนบุคคล

นายนวพงศ์	อารีกุล	54010685
นายกันตพัฒน์	เลิศอำไพกุลวงศ์	55010071
รศ. เจริญ	วงษ์ชุ่มเย็น	อาจารย์ที่ปรึกษา
ดร. ปกรณ์	วัฒนจตุรพร	อาจารย์ที่ปรึกษาร่วม

ปีการศึกษา 2558

บทคัดย่อ

โครงงานนี้จัดทำเพื่อศึกษาออกแบบ และพัฒนาแอพพลิเคชั่นสำหรับการวางแผนการเงินส่วนบุคคลเพื่อให้บุคคลทั่วไปที่สนใจด้านการวางแผนการเงินส่วนบุคคลสามารถจัดทำงบประมาณส่วนบุคคลได้และเพื่อให้บรรลุเป้าหมายการเงินที่วางแผนเอาไว้ได้ แอพพลิเคชั่นนี้พัฒนาให้รองรับกับระบบ iOS สำหรับ iPhone ซึ่งในแอพพลิเคชั่นนั้นจะมีตั้งแต่การตั้งเป้าหมายการเงินแล้วคำนวณเป็นงบประมาณที่ต้องออมเพื่อบรรลุเป้าหมายนั้นรวมถึงการจัดทำงบการเงินส่วนบุคคลเพื่อที่รวบรวมข้อมูลทางการเงินไว้ในที่เดียว เพื่อที่จะสามารถวิเคราะห์ในด้านการเงินได้ เช่น สถานะการเงิน ที่มาของเงิน และการเงินของเราทั้งหมด

Personal Finance Application

Mr. Nawapong	Areekul	54010685
Mr. Kantaphat	Lertampaisakulwong	55010071
Asst.Prof. Charoen	Vongchumyen	Advisor
Dr. Pakorn	Watanachaturaporn	Co-Advisor

Academic Year 2015

ABSTRACT

This project is to design and develop an application for personal financial planning. The users who are interested in personal financial planning, so they can plan the budget accordingly.

The application is executed for iPhone. Users first set up their usage budgets and targeted saving amount. The application collet usage financial information and planned usage in one place. Then, financial activities can be analysed including financial statement, incomes and expense.

Personal Finance Application

Mr. Nawapong	Areekul	54010685
Mr. Kantaphat	Lertampaisakulwong	55010071
Asst.Prof. Charoen	Vongchumyen	Advisor
Dr. Pakorn	Watanachaturaporn	Co-Advisor

Academic Year 2015

ABSTRACT

This project is to design and develop an application for personal financial planning. The users who are interested in personal financial planning, so they can plan the budget accordingly.

The application is executed for iPhone. Users first set up their usage budgets and targeted saving amount. The application collet usage financial information and planned usage in one place. Then, financial activities can be analysed including financial statement, incomes and expense.

กิตติกรรมประกาศ

กิตติกรรมประกาศ เป็นการกล่าวขอบคุณบุคคลที่มีส่วนร่วมให้ความช่วยเหลือให้ปริญญานิพนธ์นี้สำเร็จลงได้ด้วยดี อันดับแรกขอกล่าวขอบพระคุณ รศ. เจริญ วงษ์ชุ่มเย็น ซึ่งเป็นอาจารย์ที่ปรึกษา ที่คอยดูแลและชี้แนะแนวทางตลอดเส้นทางจนปริญญานิพนธ์นี้สำเร็จไปได้ด้วยดี รวมถึง ดร.ปกรณ์ วัฒนจตุรพร อาจารย์ที่ปรึกษาร่วม ทั้งนี้ต้องขอขอบคุณท่านทั้งสองที่ให้แนวคิดและวิธีการทำงานที่สามารถนำมาประยุกต์ใช้ในการทำงานในอนาคตได้ตลอดจนคำแนะนำตลอดเส้นทางทำให้ปริญญานิพนธ์นี้ดียิ่งขึ้นไป ถ้าหากปราศจากสองท่านนี้แล้วปริญญานิพนธ์นี้อาจจะไม่สำเร็จลุล่วงได้

ขอขอบคุณคณะวิศวกรรมคอมพิวเตอร์รวมถึงคณาจารย์ที่สอนศาสตร์วิชาความรู้ทั้งด้านวิชาการและจริยธรรม ตลอดเส้นทางในการศึกษาในสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังแห่งนี้ ที่หล่อหลอมจากนักศึกษาให้ก้าวขึ้นมาเป็นบุคคลากรที่สำคัญในการพัฒนาประเทศชาติและสามารถนำวิชาความรู้ที่ได้ไปใช้ประกอบวิชาชีพในอนาคต

สุดท้ายนี้จะขอกล่าวขอบพระคุณบิดา มารดาที่คอยเป็นกำลังใจ ตลอดเส้นทางและทำให้ได้มีโอกาสเข้ามาศึกษาในสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังแห่งนี้ทำให้ได้เจอเพื่อน รุ่นพี่และรุ่นน้องที่น่ารัก ที่คอยช่วยเหลือเกื้อกูลดูแลและได้เรียนรู้ประสบการณ์ในการทำงานร่วมกัน ซึ่งเป็นอีกส่วนหนึ่งที่ทำให้ปริญญานิพนธ์นี้สำเร็จลุล่วงไปได้ด้วยดี

นวพงศ์ อารีกุล
กันตพัฒน์ เลิศอำไพกุลวงศ์

กิตติกรรมประกาศ

กิตติกรรมประกาศ เป็นการกล่าวขอบคุณบุคคลที่มีส่วนร่วมให้ความช่วยเหลือให้ปริญญานิพนธ์นี้สำเร็จลงได้ด้วยดี อันดับแรกขอกล่าวขอบพระคุณ รศ. เจริญ วงษ์ชุ่มเย็น ซึ่งเป็นอาจารย์ที่ปรึกษา ที่คอยดูแลและชี้แนะแนวทางตลอดเส้นทางจนปริญญานิพนธ์นี้สำเร็จไปได้ด้วยดี รวมถึง ดร.ปกรณ์ วัฒนจตุรพร อาจารย์ที่ปรึกษาร่วม ทั้งนี้ต้องขอขอบคุณท่านทั้งสองที่ให้แนวคิดและวิธีการทำงานที่สามารถนำมาประยุกต์ใช้ในการทำงานในอนาคตได้ตลอดจนคำแนะนำตลอดเส้นทางทำให้ปริญญานิพนธ์นี้ดียิ่งขึ้นไป ถ้าหากปราศจากสองท่านนี้แล้วปริญญานิพนธ์นี้อาจจะไม่สำเร็จลุล่วงได้

ขอขอบคุณคณะวิศวกรรมคอมพิวเตอร์รวมถึงคณาจารย์ที่สอนศาสตร์วิชาความรู้ทั้งด้านวิชาการและจริยธรรม ตลอดเส้นทางในการศึกษาในสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังแห่งนี้ ที่หล่อหลอมจากนักศึกษาให้ก้าวขึ้นมาเป็นบุคคลากรที่สำคัญในการพัฒนาประเทศชาติและสามารถนำวิชาความรู้ที่ได้ไปใช้ประกอบวิชาชีพในอนาคต

สุดท้ายนี้จะขอกล่าวขอบพระคุณบิดา มารดาที่คอยเป็นกำลังใจ ตลอดเส้นทางและทำให้ได้มีโอกาสเข้ามาศึกษาในสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังแห่งนี้ทำให้ได้เจอเพื่อน รุ่นพี่และรุ่นน้องที่น่ารัก ที่คอยช่วยเหลือเกื้อกูลดูแลและได้เรียนรู้ประสบการณ์ในการทำงานร่วมกัน ซึ่งเป็นอีกส่วนหนึ่งที่ทำให้ปริญญานิพนธ์นี้สำเร็จลุล่วงไปได้ด้วยดี

นวพงศ์ อารีกุล

กันตพัฒน์ เลิศอำไพกุลวงศ์

สารบัญ

หน้า

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตาราง **หน้า**

สารบัญตาราง

ตาราง **หน้า**

สารบัญรูป

รูป **หน้า**

สารบัญรูป

รูป **หน้า**

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

รูป **หน้า**

สารบัญรูป (ต่อ)

รูป **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและปัญหา

ในปัจจุบันผู้คนเริ่มมีการสนใจด้านการเงินและการลงทุนกันมากขึ้นซึ่งสิ่งที่เป็นพื้นฐานในเรื่องการเงินส่วนบุคคลก็คือ การทำรายรับ-รายจ่าย แต่บางทีพบปัญหาว่าไม่สะดวกที่จะทำรายรับรายจ่ายในชีวิตประจำวันประกอบกับในปัจจุบันมีการใช้ Smart Phone เป็นส่วนหนึ่งในชีวิตประจำวัน ทำให้การทำรายรับจ่ายนั้นง่ายขึ้นเนื่องจากสามารถบันทึกลงไปในแอพใน Smart Phone ได้เลย

ประกอบกับทางผู้จัดทำได้มีโอกาสได้ไปศึกษาเกี่ยวกับการวางแผนการเงินกับทางตลาดหลักทรัพย์และได้เห็นความสำคัญเรื่องการวางแผนทางการเงิน อย่างเรื่องงบการเงินส่วนบุคคลทั้งงบดุลงบกระแสเงินสดจึงต้องการนำความรู้ที่ได้ตรงนี้มาพัฒนาต่อยอดโดยนำความรู้ทางด้านการเงินและด้านวิศวกรรมคอมพิวเตอร์เข้ามาประยุกต์ใช้ร่วมกัน

โดยโปรแกรมถูกออกแบบให้สามารถแก้ปัญหาที่ไม่สามารถคำนวนการเก็บเงินเพื่อเป้าหมายในชีวิตต่างๆ ออกมาเป็นเงินออมในแต่ละเดือนได้ สามารถรู้สถานะการเงินของตัวเองได้ รวมทั้งการวางแผนทางภาษีได้

1.2 วัตถุประสงค์ของโครงงาน

1) เพื่อนำความรู้ด้านการเงินและวิศวกรรมคอมพิวเตอร์มาประยุกต์ใช้เข้าด้วยกัน
2) เพื่อให้ผู้ใช้งานมีความรู้ในการวางแผนการเงินส่วนบุคคลมากขึ้น
3) เพื่อศึกษาขีดความสามารถของ Swift Language
4) เพื่อประยุกต์รูปแบบการวางแผนการเงินให้ง่ายมากขึ้นในแง่การใช้งานบนสมาร์ทโฟน (iOS) เป็นสื่อใช้งานเพื่อความสะดวกสบาย
5) เพื่อให้ผู้ใช้งานสามารถวิเคราะห์ข้อมูลวางแผนการเงินโดยสรุปเป็นรูปแบบกราฟตามที่ผู้ใช้ต้องการ
6) เพื่อให้สามารถนำข้อมูล อดีต ปัจจุบัน อนาคต มาประกอบวิเคราะห์ การออมเงินในแต่ละครั้ง (ต่อเดือน)

บทที่ 1
บทนำ

1.1 ความเป็นมาและปัญหา

ในปัจจุบันผู้คนเริ่มมีการสนใจด้านการเงินและการลงทุนกันมากขึ้นซึ่งสิ่งที่เป็นพื้นฐานในเรื่องการเงินส่วนบุคคลก็คือ การทำรายรับ-รายจ่าย แต่บางทีพบปัญหาว่าไม่สะดวกที่จะทำรายรับรายจ่ายในชีวิตประจำวันประกอบกับในปัจจุบันมีการใช้ Smart Phone เป็นส่วนหนึ่งในชีวิตประจำวัน ทำให้การทำรายรับจ่ายนั้นง่ายขึ้นเนื่องจากสามารถบันทึกลงไปในแอพใน Smart Phone ได้เลย

ประกอบกับทางผู้จัดทำได้มีโอกาสได้ไปศึกษาเกี่ยวกับการวางแผนการเงินกับทางตลาดหลักทรัพย์และได้เห็นความสำคัญเรื่องการวางแผนทางการเงิน อย่างเรื่องงบการเงินส่วนบุคคลทั้งงบดุลงบกระแสเงินสดจึงต้องการนำความรู้ที่ได้ตรงนี้มาพัฒนาต่อยอดโดยนำความรู้ทางด้านการเงินและด้านวิศวกรรมคอมพิวเตอร์เข้ามาประยุกต์ใช้ร่วมกัน

โดยโปรแกรมถูกออกแบบให้สามารถแก้ปัญหาที่ไม่สามารถคำนวนการเก็บเงินเพื่อเป้าหมายในชีวิตต่างๆ ออกมาเป็นเงินออมในแต่ละเดือนได้ สามารถรู้สถานะการเงินของตัวเองได้ รวมทั้งการวางแผนทางภาษีได้

1.2 วัตถุประสงค์ของโครงงาน

1) เพื่อนำความรู้ด้านการเงินและวิศวกรรมคอมพิวเตอร์มาประยุกต์ใช้เข้าด้วยกัน
2) เพื่อให้ผู้ใช้งานมีความรู้ในการวางแผนการเงินส่วนบุคคลมากขึ้น
3) เพื่อศึกษาขีดความสามารถของ Swift Language
4) เพื่อประยุกต์รูปแบบการวางแผนการเงินให้ง่ายมากขึ้นในแง่การใช้งานบนสมาร์ทโฟน (iOS) เป็นสื่อใช้งานเพื่อความสะดวกสบาย
5) เพื่อให้ผู้ใช้งานสามารถวิเคราะห์ข้อมูลวางแผนการเงินโดยสรุปเป็นรูปแบบกราฟตามที่ผู้ใช้ต้องการ
6) เพื่อให้สามารถนำข้อมูล อดีต ปัจจุบัน อนาคต มาประกอบวิเคราะห์ การออมเงินในแต่ละครั้ง (ต่อเดือน)

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้นำความรู้ทางวิศวกรรมคอมพิวเตอร์มาประยุกต์ใช้ในการทำงาน

2) ได้ความรู้ด้านการเงินส่วนบุคคลและสามารถนำไปใช้ในชีวิตประจำวันได้

3) ได้รับความรู้ ความเข้าใจเกี่ยวกับขีดความสามารถของ Swift Language

4) ได้รับความรู้ ความเข้าใจเกี่ยวกับการวางแผนการเงินเป็นรูปแบบที่ดูเข้าใจง่าย

5) สามารถนำความรู้ที่ได้ นำไปออกแบบและพัฒนาแอพพลิเคชั่นตามที่ได้ออกแบบ

6) สามารถนำแอพพลิเคชั่นที่พัฒนาขึ้นไปใช้งานที่เกี่ยวข้องกับการวางแผนการเงินส่วนบุคคล

1.4 ขอบเขตของโครงงาน

1) พัฒนาแอพพลิเคชั่น การเงินส่วนบุคคล สำหรับระบบ iOS เฉพาะ iPhone

2) พัฒนาแอพพลิเคชั่นโดยใช้ภาษา Swift

3) ใช้หลักการวางแผนการเงินส่วนบุคคลโดยอ้างอิงตามหลัก Certified Financial Planner

1.4.1 ความสามารถของโปรแกรม

แอพพลิเคชั่นการเงินส่วนบุคคลทำงานผ่าน iOS (iPhone) จะแสดงภาพรวมการวางแผนการเงินส่วนบุคคลของผู้ใช้ โดยแสดงเป็นรูปแบบกราฟและแสดงผลลัพธ์เป็นรูปแบบที่ผู้ใช้ทั่วไปที่ไม่ยังไม่เข้าใจเกี่ยวกับการเงินทั้งหมดออกมาดูง่าย เข้าใจง่าย โดยทั้งหมดต้องเริ่มต้นจากผู้ใช้ต้องป้อนค่าผ่านแอพพลิเคชั่น ซึ่งตัวแอพพลิเคชั่นจะแสดงผลลัพธ์ตามที่ผู้ใช้ต้องการ โดยบนตัวแอพพลิเคชั่นมีฟังก์ชั่นการใช้งานหลัก ดังนี้

1.4.1.1 มีการตั้งเป้าหมายทางการเงิน

ประกอบไปด้วย

1) เป้าหมาย เช่น วางแผน ทุนการศึกษาให้ลูก เงินดาวน์บ้านหรือรถเป็นต้น

2) ระยะเวลา ที่ต้องการเก็บออม

3) จำนวนเงิน ที่ใช้ในเป้าหมายนั้น ซึ่งจะนำมาคำนวณเพื่อเป็นเงินออมที่จะเก็บต่อเดือนได้

1.4.1.2 มีการตรวจสุขภาพทางการเงิน

โดยคำนวณจากงบการเงินส่วนบุคคลและวิเคราะห์เพื่อปรับปรุงการเงินของผู้ใช้ให้ดีขึ้น

1.4.1.3 คำนวนการลดหย่อนภาษี

ช่วยทางแผนภาษีให้บุคคลธรรมดาในลดหย่อน LFT, RMF และเบี้ยประกันแบบบำนาญ

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้นำความรู้ทางวิศวกรรมคอมพิวเตอร์มาประยุกต์ใช้ในการทำงาน

2) ได้ความรู้ด้านการเงินส่วนบุคคลและสามารถนำไปใช้ในชีวิตประจำวันได้

3) ได้รับความรู้ ความเข้าใจเกี่ยวกับขีดความสามารถของ Swift Language

4) ได้รับความรู้ ความเข้าใจเกี่ยวกับการวางแผนการเงินเป็นรูปแบบที่ดูเข้าใจง่าย

5) สามารถนำความรู้ที่ได้ นำไปออกแบบและพัฒนาแอพพลิเคชั่นตามที่ได้ออกแบบ

6) สามารถนำแอพพลิเคชั่นที่พัฒนาขึ้นไปใช้งานที่เกี่ยวข้องกับการวางแผนการเงินส่วนบุคคล

1.4 ขอบเขตของโครงงาน

1) พัฒนาแอพพลิเคชั่น การเงินส่วนบุคคล สำหรับระบบ iOS เฉพาะ iPhone

2) พัฒนาแอพพลิเคชั่นโดยใช้ภาษา Swift

3) ใช้หลักการวางแผนการเงินส่วนบุคคลโดยอ้างอิงตามหลัก Certified Financial Planner

1.4.1 ความสามารถของโปรแกรม

แอพพลิเคชั่นการเงินส่วนบุคคลทำงานผ่าน iOS (iPhone) จะแสดงภาพรวมการวางแผนการเงินส่วนบุคคลของผู้ใช้ โดยแสดงเป็นรูปแบบกราฟและแสดงผลลัพธ์เป็นรูปแบบที่ผู้ใช้ทั่วไปที่ไม่ยังไม่เข้าใจเกี่ยวกับการเงินทั้งหมดออกมาดูง่าย เข้าใจง่าย โดยทั้งหมดต้องเริ่มต้นจากผู้ใช้ต้องป้อนค่าผ่านแอพพลิเคชั่น ซึ่งตัวแอพพลิเคชั่นจะแสดงผลลัพธ์ตามที่ผู้ใช้ต้องการ โดยบนตัวแอพพลิเคชั่นมีฟังก์ชั่นการใช้งานหลัก ดังนี้

1.4.1.1 มีการตั้งเป้าหมายทางการเงิน

ประกอบไปด้วย

1) เป้าหมาย เช่น วางแผน ทุนการศึกษาให้ลูก เงินดาวน์บ้านหรือรถเป็นต้น

2) ระยะเวลา ที่ต้องการเก็บออม

3) จำนวนเงิน ที่ใช้ในเป้าหมายนั้น ซึ่งจะนำมาคำนวณเพื่อเป็นเงินออมที่จะเก็บต่อเดือนได้

1.4.1.2 มีการตรวจสุขภาพทางการเงิน

โดยคำนวณจากงบการเงินส่วนบุคคลและวิเคราะห์เพื่อปรับปรุงการเงินของผู้ใช้ให้ดีขึ้น

1.4.1.3 คำนวนการลดหย่อนภาษี

ช่วยทางแผนภาษีให้บุคคลธรรมดาในลดหย่อน LFT, RMF และเบี้ยประกันแบบบำนาญ

1.5 คุณสมบัติของโปรแกรม

ตาราง 1.1 Check-List คุณสมบัติของโปรแกรม

Feature	รายละเอียด	Check
1. ตรวจสุขภาพการเงิน		
1.1 วิเคราะห์อัตราส่วนสินทรัพย์ที่มีสภาพคล่องต่อความมั่งคั่งสุทธิ	สัดส่วนสภาพคล่องอย่างเช่น กรณี ฉุกเฉิน	
1.2 วิเคราะห์อัตราส่วนหนี้สินต่อสินทรัพย์	ความสามารถในการชำระหนี้สินในอนาคต	
1.3 วิเคราะห์อัตราแสดงการชำระคืนหนี้สินจากรายได้	ความสามารถในการชำระหนี้สินต่อรายได้	
1.4 วิเคราะห์อัตราส่วนการชำระคืนหนี้สินที่ไม่ใช่การจำนองรายได้	ความสามารถในการชำระหนี้สินยกเว้น บ้าน รถ	
1.5 วิเคราะห์อัตราส่วนการออม	แสดงถึงสัดส่วนการออมที่บุคคลกันไว้จากรายได้รวม	
1.6 วิเคราะห์อัตราส่วยการลงทุน	แสดงถึงสินทรัพย์ที่บุคคลลงทุนโดยมุ่งหวังผลตอบแทน	
1.7 ความมั่งคั่งสุทธิ	ทรัพย์สินทั้งหมดลบด้วยหนี้สิน	
2. เป้าหมายทางการเงิน		
2.1 คำนวณเป้าหมายออกมาเป็นการปฏิบัติ	คำนวณเป้าหมายออกมาเป็นการจัดงบประมาณรายเดือนเพื่อให้ง่ายต่อการนำไปปฏิบัติ	
2.2 มีการเลือกรูปแบบในการออม	จะนำรูปแบบมาคำนวณอัตราผลตอบแทนในการออม	

1.5 คุณสมบัติของโปรแกรม

ตาราง 1.1 Check-List คุณสมบัติของโปรแกรม

Features	รายละเอียด	Check
1. ตรวจสุขภาพการเงิน		
1.1 วิเคราะห์อัตราส่วนสินทรัพย์ที่มีสภาพคล่องต่อความมั่งคั่งสุทธิ	สัดส่วนสภาพคล่องอย่างเช่น กรณี ฉุกเฉิน	
1.2 วิเคราะห์อัตราส่วนหนี้สินต่อสินทรัพย์	ความสามารถในการชำระหนี้สินในอนาคต	
1.3 วิเคราะห์อัตราแสดงการชำระคืนหนี้สินจากรายได้	ความสามารถในการชำระหนี้สินต่อรายได้	
1.4 วิเคราะห์อัตราส่วนการชำระคืนหนี้สินที่ไม่ใช่การจำนองรายได้	ความสามารถในการชำระหนี้สินยกเว้น บ้าน รถ	
1.5 วิเคราะห์อัตราส่วนการออม	แสดงถึงสัดส่วนการออมที่บุคคลกันไว้จากรายได้รวม	
1.6 วิเคราะห์อัตราส่วยการลงทุน	แสดงถึงสินทรัพย์ที่บุคคลลงทุนโดยมุ่งหวังผลตอบแทน	
1.7 ความมั่งคั่งสุทธิ	ทรัพย์สินทั้งหมดลบด้วยหนี้สิน	
2. เป้าหมายทางการเงิน		
2.1 คำนวณเป้าหมายออกมาเป็นการปฏิบัติ	คำนวณเป้าหมายออกมาเป็นการจัดงบประมาณรายเดือนเพื่อให้ง่ายต่อการนำไปปฏิบัติ	
2.2 มีการเลือกรูปแบบในการออม	จะนำรูปแบบมาคำนวณอัตราผลตอบแทนในการออม	

ตาราง 1.1 Check-List คุณสมบัติของโปรแกรม (ต่อ)

Feature	รายละเอียด	Check
3. ภาษี		
3.1 คำนวณลดหย่อน LTF	จำนวนลดหย่อนสูงสุดของ LTF ที่สามารถลดหย่อนได้	
3.2 ค่าลดหย่อน RMF	จำนวนลดหย่อนสูงสุดของ RMF ที่สามารถลดหย่อนได้	
3.3 คำนวณลดหย่อนประกันชีวิตแบบบำนาญ	จำนวนลดหย่อนสูงสุดของเบี้ยประกันแบบบำนาญ ที่สามารถลดหย่อนได้	

1.6 วิธีการดำเนินงาน

1) ศึกษาทฤษฎีของ Swift Language
2) ศึกษาการใช้งาน Xcode Developer Tools
3) ศึกษาทฤษฎีการออกแบบและการใช้งานฐานข้อมูล (Realm Database)
4) ศึกษาอัลกอริทึมในการสรุปข้อมูลทั้งหมดเพื่อประมวลผลให้ผู้ใช้งานเข้าใจง่าย
5) ทดลองเขียนโปรแกรมภาษา Swift ให้เข้าใจเบื้องต้น
6) วิเคราะห์และออกแบบแอพพลิเคชั่นที่จะสร้าง
7) พัฒนาแอพพลิเคชั่นจากการออกแบบสำหรับ iOS (iPhone)
8) ทดสอบประสิทธิภาพและแก้ไขระบบ
9) สรุปผล และจัดทำรายงานรูปเล่ม

ตาราง 1.1 Check-List คุณสมบัติของโปรแกรม (ต่อ)

[illegible]	รายละเอียด	Check
3. ภาษี		
3.1 คำนวณลดหย่อน LTF	จำนวนลดหย่อนสูงสุดของ LTF ที่สามารถลดหย่อนได้	
3.2 ค่าลดหย่อน RMF	จำนวนลดหย่อนสูงสุดของ RMF ที่สามารถลดหย่อนได้	
3.3 คำนวณลดหย่อนประกันชีวิตแบบบำนาญ	จำนวนลดหย่อนสูงสุดของเบี้ยประกันแบบบำนาญ ที่สามารถลดหย่อนได้	

1.6 วิธีการดำเนินงาน

1) ศึกษาทฤษฎีของ Swift Language
2) ศึกษาการใช้งาน Xcode Developer Tools
3) ศึกษาทฤษฎีการออกแบบและการใช้งานฐานข้อมูล (Realm Database)
4) ศึกษาอัลกอริทึ่มในการสรุปข้อมูลทั้งหมดเพื่อประมวลผลให้ผู้ใช้งานเข้าใจง่าย
5) ทดลองเขียนโปรแกรมภาษา Swift ให้เข้าใจเบื้องต้น
6) วิเคราะห์และออกแบบแอพพลิเคชั่นที่จะสร้าง
7) พัฒนาแอพพลิเคชั่นจากการออกแบบสำหรับ iOS (iPhone)
8) ทดสอบประสิทธิภาพและแก้ไขระบบ
9) สรุปผล และจัดทำรายงานรูปเล่ม

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

ทฤษฎีที่นำมาใช้ในการพัฒนาแอพพลิเคชั่นนั้นในบทนี้จะกล่าวถึงด้วยกัน 3 เรื่องคือ ทฤษฎีเบื้องต้นเกี่ยวการวางแผนการเงินส่วนบุคคลซึ่งจะอ้างอิงตามหลัก (CFP), ทฤษฎีเบื้องต้นเกี่ยวกับระบบ iOS ซึ่งจะกล่าวถึงระบบ iOS และสถาปัตยกรรมของระบบ และส่วนสุดท้ายคือภาษา Swift ซึ่งเป็นภาษาใหม่ในการพัฒนาแอพพลิเคชั่นของระบบ iOS

2.1 ทฤษฎีเบื้องต้นเกี่ยวกับการวางแผนการเงินส่วนบุคคล

2.1.1 CFP คืออะไร

CFP ย่อมาจาก **Certified Financial Planner** เป็นคุณวุฒิวิชาชีพสำหรับนักวางแผนการเงินที่ได้รับการยอมรับอย่างกว้างขวาง แสดงถึงการมีความรู้ ความสามารถ ทั้งภาคทฤษฎีและปฏิบัติ ในการให้คำแนะนำและวางแผนการเงินบุคคลแก่ลูกค้าครอบคลุมในทุก ด้าน ได้แก่ ***การวางแผนการลงทุน การวางแผนประกันชีวิต การวางแผนเพื่อการเกษียณ การวางแผนภาษีและมรดก*** และบูรณาการแผนการเงินในทุกด้านเข้ามาไว้ด้วยกันเป็น***แผนการเงินฉบับสมบูรณ์*** และให้ลูกค้าได้รับประโยชน์สูงสุดจากการวางแผนการเงิน โดยคำนึงถึงเป้าหมายทางการเงินของลูกค้าเป็นสำคัญ

ปัจจุบันทั่วโลกมีนักวางแผนการเงินที่มีคุณวุฒิ CFP จำนวนมากกว่าหนึ่งแสนคน สำหรับ***สมาคมนักวางแผนการเงินไทย (Thai Financial Planners Association: TFPA)*** เป็นหน่วยงานที่ทำหน้าที่ในการรับรองคุณวุฒิ CFP ให้แก่นักวางแผนการเงินของไทยที่มีคุณสมบัติครบตามมาตรฐานสากล โดยนักวางแผนการเงินจะต้องมีสมรรถภาพที่ครอบคลุมใน 3 ด้านสำคัญ ประกอบด้วย ความสามารถ (Ability), ทักษะ (Skill), และความรู้ (Knowledge) ซึ่งคุณสมบัติเหล่านี้จะสอดคล้องกับกฎกติกา กฎหมายของแต่ละประเทศ เพราะ CFP มุ่งหวังให้เกิดการวางแผนการเงินจริงแต่บุคคล หรือกลุ่มบุคคล จึงจำเป็นต้องเข้าในระบบกฎหมาย ระเบียบ วิธีการ เงื่อนไขที่แตกต่างกันของแต่ละประเทศ เช่น กฎหมายด้านประกัน การลงทุน ภาษี กองทุนเพื่อการเกษียณ เป็นต้น

2.1.2 เป้าหมายทางการเงิน

เป้าหมายการเงินที่ดีต้องมีองค์ประกอบสำคัญ 3 ประการเพื่อให้สามารถนำมาเป็นแผนปฏิบัติได้ ดังต่อไปนี้คือ

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

ทฤษฎีที่นำมาใช้ในการพัฒนาแอพพลิเคชั่นนั้นในบทนี้จะกล่าวถึงด้วยกัน 3 เรื่องคือ ทฤษฎีเบื้องต้นเกี่ยวการวางแผนการเงินส่วนบุคคลซึ่งจะอ้างอิงตามหลัก (CFP), ทฤษฎีเบื้องต้นเกี่ยวกับระบบ iOS ซึ่งจะกล่าวถึงระบบ iOS และสถาปัตยกรรมของระบบ และส่วนสุดท้ายคือภาษา Swift ซึ่งเป็นภาษาใหม่ในการพัฒนาแอพพลิเคชั่นของระบบ iOS

2.1 ทฤษฎีเบื้องต้นเกี่ยวกับการวางแผนการเงินส่วนบุคคล

2.1.1 CFP คืออะไร

CFP ย่อมาจาก **Certified Financial Planner** เป็นคุณวุฒิวิชาชีพสำหรับนักวางแผนการเงินที่ได้รับการยอมรับอย่างกว้างขวาง แสดงถึงการมีความรู้ ความสามารถ ทั้งภาคทฤษฎีและปฏิบัติ ในการให้คำแนะนำและวางแผนการเงินบุคคลแก่ลูกค้าครอบคลุมในทุก ด้าน ได้แก่ ***การวางแผนการลงทุน การวางแผนประกันชีวิต การวางแผนเพื่อการเกษียณ การวางแผนภาษีและมรดก*** และบูรณาการแผนการเงินในทุกด้านเข้ามาไว้ด้วยกันเป็น***แผนการเงินฉบับสมบูรณ์*** และให้ลูกค้าได้รับประโยชน์สูงสุดจากการวางแผนการเงิน โดยคำนึงถึงเป้าหมายทางการเงินของลูกค้าเป็นสำคัญ

ปัจจุบันทั่วโลกมีนักวางแผนการเงินที่มีคุณวุฒิ CFP จำนวนมากกว่าหนึ่งแสนคน สำหรับ***สมาคมนักวางแผนการเงินไทย (Thai Financial Planners Association: TFPA)*** เป็นหน่วยงานที่ทำหน้าที่ในการรับรองคุณวุฒิ CFP ให้แก่นักวางแผนการเงินของไทยที่มีคุณสมบัติครบตามมาตรฐานสากล โดยนักวางแผนการเงินจะต้องมีสมรรถภาพที่ครอบคลุมใน 3 ด้านสำคัญ ประกอบด้วย ความสามารถ (Ability), ทักษะ (Skill), และความรู้ (Knowledge) ซึ่งคุณสมบัติเหล่านี้จะสอดคล้องกับกฎกติกา กฎหมายของแต่ละประเทศ เพราะ CFP มุ่งหวังให้เกิดการวางแผนการเงินจริงแต่บุคคล หรือกลุ่มบุคคล จึงจำเป็นต้องเข้าในระบบกฎหมาย ระเบียบ วิธีการ เงื่อนไขที่แตกต่างกันของแต่ละประเทศ เช่น กฎหมายด้านประกัน การลงทุน ภาษี กองทุนเพื่อการเกษียณ เป็นต้น

2.1.2 เป้าหมายทางการเงิน

เป้าหมายการเงินที่ดีต้องมีองค์ประกอบสำคัญ 3 ประการเพื่อให้สามารถนำมาเป็นแผนปฏิบัติได้ ดังต่อไปนี้คือ

1) **อะไร (What)** เป้าหมายทางการเงินที่ดีควรเป็นรูปธรรม เช่น ต้องการมีรถยนต์ ต้องการมีบ้าน ทุนการศึกษา

2) **เท่าใด (How Much)** จากที่กำหนดเป้าหมายว่าคืออะไรแล้ว ให้กำหนดจำนวนเงิน เพื่อให้มีรายละเอียดมากขึ้นและสามารถนำมาวางแผนการเงินได้

3) **เมื่อใด (When)** เมื่อมีกำหนดเป้าหมายแล้วจำเป็นต้องใส่ระยะเวลากำหนด ถ้าไม่มีการกำหนดระยะเวลาเป้าหมายนั้นก็จะเลื่อนลอยและสามารถเลื่อนออกไปได้ตลอดเวลา

2.1.3 มูลค่าเงินตามการเวลา

ในเรื่องการวางแผนการเงินจะมีเรื่องมูลค่าเงินตามการเวลามาเกี่ยวข้องในการคำนวณ เนื่องจากมีความเกี่ยวข้องกับระยะเวลาและเรื่องของดอกเบี้ย ซึ่งอย่างเรื่องใกล้ๆ ตัวก็เห็นได้จากเรื่องของเงินเฟ้อที่ทำให้ข้าวของราคาแพงขึ้นและเมื่อในการวางแผนการเงินในระยะยาว เช่น 10 ปีขึ้นไปทำให้ค่าเงินตรงนี้เปลี่ยนไปมากและได้นำสูตรสมการในการคำนวณมาใช้ ในโปรแกรม

$$PV(1+i)^n + PMT\left[\frac{(1+i)^n - 1}{i}\right] + FV = 0 \qquad (2.1)$$

2.1.4 อัตราผลตอบแทนที่นำมาอ้างอิงในการคำนวณ

ประเภทสินทรัพย์	อัตราผลตอบแทนคาดหวังของสินทรัพย์
ตราสารทุน	12.0%
ตราสารหนี้	5.7%
เงินฝาก	1.0%

ประเภทสินทรัพย์	เกณฑ์มาตรฐาน (Benchmark)
เงินฝาก	อัตราดอกเบี้ยเงินฝากประจำ 12 เดือน เฉลี่ยของ BBL, KBANK, KTB, SCB แหล่งข้อมูล: www.bot.or.th
ตราสารหนี้	อัตราผลตอบแทนของพันธบัตร รัฐบาลที่มีอายุ 1, 5, 10 และ 20 ปี แหล่งข้อมูล: www.thaibma.or.th
ตราสารทุน	อัตราผลตอบแทนของ SET Index หรือ SET50 Index แหล่งข้อมูล: www.set.or.th

รูป 2.1 อัตราผลตอบแทนของสินทรัพย์ต่างๆ

1) **อะไร (What)** เป้าหมายทางการเงินที่ดีควรเป็นรูปธรรม เช่น ต้องการมีรถยนต์ ต้องการมีบ้าน ทุนการศึกษา

2) **เท่าใด (How Much)** จากที่กำหนดเป้าหมายว่าคืออะไรแล้ว ให้กำหนดจำนวนเงิน เพื่อให้มีรายละเอียดมากขึ้นและสามารถนำมาวางแผนการเงินได้

3) **เมื่อใด (When)** เมื่อมีกำหนดเป้าหมายแล้วจำเป็นต้องใส่ระยะเวลากำหนด ถ้าไม่มีการกำหนดระยะเวลาเป้าหมายนั้นก็จะเลื่อนลอยและสามารถเลื่อนออกไปได้ตลอดเวลา

2.1.3 มูลค่าเงินตามการเวลา

ในเรื่องการวางแผนการเงินจะมีเรื่องมูลค่าเงินตามการเวลามาเกี่ยวข้องในการคำนวณ เนื่องจากมีความเกี่ยวข้องกับระยะเวลาและเรื่องของดอกเบี้ย ซึ่งอย่างเรื่องใกล้ๆ ตัวก็เห็นได้จากเรื่องของเงินเฟ้อที่ทำให้ข้าวของราคาแพงขึ้นและเมื่อในการวางแผนการเงินในระยะยาว เช่น 10 ปีขึ้นไปทำให้ค่าเงินตรงนี้เปลี่ยนไปมากและได้นำสูตรสมการในการคำนวณมาใช้ ในโปรแกรม

$$PV(1+i)^n + PMT\left[\frac{(1+i)^n - 1}{i}\right] + FV = 0 \qquad (2.1)$$

2.1.4 อัตราผลตอบแทนที่นำมาอ้างอิงในการคำนวณ

ประเภทสินทรัพย์	อัตราผลตอบแทนคาดหวังของสินทรัพย์
ตราสารทุน	12.0%
ตราสารหนี้	5.7%
เงินฝาก	1.0%

ประเภทสินทรัพย์	เกณฑ์มาตรฐาน (Benchmark)
เงินฝาก	อัตราดอกเบี้ยเงินฝากประจำ 12 เดือน เฉลี่ยของ BBL, KBANK, KTB, SCB แหล่งข้อมูล: www.bot.or.th
ตราสารหนี้	อัตราผลตอบแทนของพันธบัตร รัฐบาลที่มีอายุ 1, 5, 10 และ 20 ปี แหล่งข้อมูล: www.thaibma.or.th
ตราสารทุน	อัตราผลตอบแทนของ SET Index หรือ SET50 Index แหล่งข้อมูล: www.set.or.th

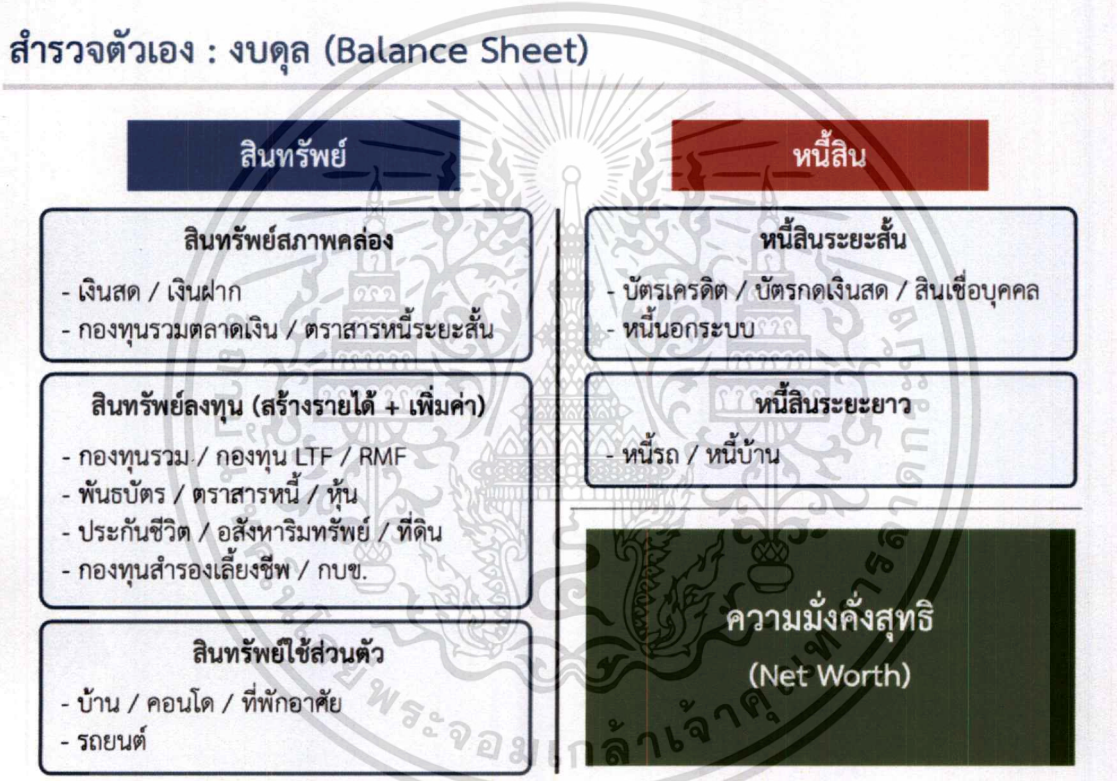
รูป 2.1 อัตราผลตอบแทนของสินทรัพย์ต่างๆ

2.1.5 งบการเงินส่วนบุคคล

งบการเงินส่วนบุคคลนั้นเป็นสิ่งแสดงถึงสถานะการเงิน การใช้จ่ายของบุคคลนั้นทำให้สามารถวิเคราะห์และวางแผนการเงินของบุคคลนั้นได้ซึ่งประกอบไปด้วยงบการเงินสองส่วนหลักด้วยกัน คือ

2.1.5.1 งบดุล

งบดุลเป็นงบแสดงสถานะทางการเงินซึ่งจะเปรียบเทียบก็คล้ายกับทรานสคริปต์ของนักศึกษาที่แสดงถึงผลการเรียนว่าดีหรือไม่ ซึ่งในด้านการเงินนั้นเราสามารถนำตัวเลขในงบดุลมาวิเคราะห์เพื่อปรับปรุงให้ดีขึ้นได้



รูป 2.2 ตัวอย่างงบดุลและประเภทสินทรัพย์ต่างๆ

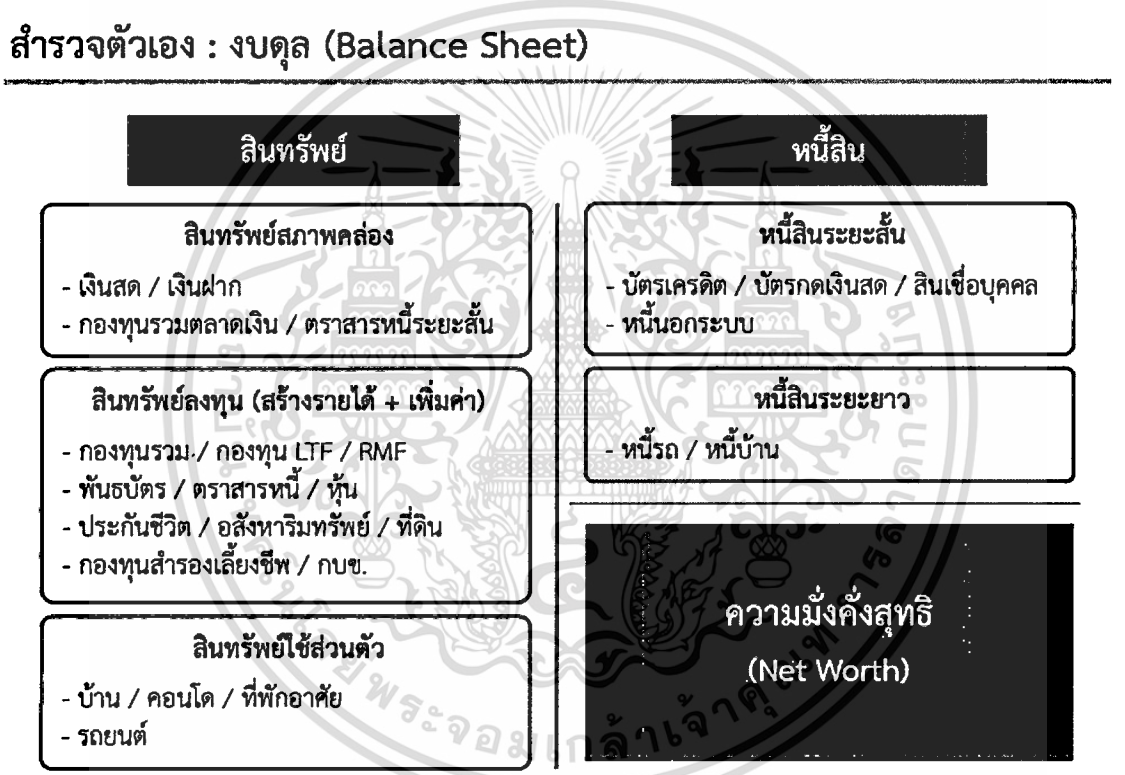
รูป 2.2 จะเห็นได้ว่ามีการแบ่งประเภทสินทรัพย์ ออกเป็นหลายประเภท ซึ่งทำให้เราสามารถวิเคราะห์สถานะการเงินของเราได้ ซึ่งแบ่งได้เป็นสองประเภทหลัก คือ ทรัพย์สินและหนี้สิน

2.1.5 งบการเงินส่วนบุคคล

งบการเงินส่วนบุคคลนั้นเป็นสิ่งแสดงถึงสถานะการเงิน การใช้จ่ายของบุคคลนั้นทำให้สามารถวิเคราะห์และวางแผนการเงินของบุคคลนั้นได้ซึ่งประกอบไปด้วยงบการเงินสองส่วนหลักด้วยกัน คือ

2.1.5.1 งบดุล

งบดุลเป็นงบแสดงสถานะทางการเงินซึ่งจะเปรียบเทียบก็คล้ายกับทรานสคริปต์ของนักศึกษาที่แสดงถึงผลการเรียนว่าดีหรือไม่ ซึ่งในด้านการเงินนั้นเราสามารถนำตัวเลขในงบดุลมาวิเคราะห์เพื่อปรับปรุงให้ดีขึ้นได้



รูป 2.2 ตัวอย่างงบดุลและประเภทสินทรัพย์ต่างๆ

รูป 2.2 จะเห็นได้ว่ามีการแบ่งประเภทสินทรัพย์ ออกเป็นหลายประเภท ซึ่งทำให้เราสามารถวิเคราะห์สถานะการเงินของเราได้ ซึ่งแบ่งได้เป็นสองประเภทหลัก คือ ทรัพย์สินและหนี้สิน

แล้วแบ่งย่อยตามระยะเวลา

1) **สินทรัพย์**

a) **สินทรัพย์สภาพคล่อง** เป็นสินทรัพย์ที่มีสภาพคล่องสูงคือสามารถนำมาใช้จ่ายได้ง่าย อย่างเช่นเงินสด สามารถนำมาใช้จ่ายได้ทันที หรือ พวกกองทุนตลาดเงิน ซึ่งสามารถขายกองทุนแล้วนำเงินมาใช้โดยใช้ระยะ 1-2 วัน เพื่อกำหนดจัดงบประมาณสำหรับการใช้จ่ายโดยทั่วไป

b) **สินทรัพย์ลงทุน** เป็นสินทรัพย์ที่มีสภาพคล่องน้อยกว่าสินทรัพย์สภาพซึ่งถ้าขายมาใช้อาจจะเกิดการขาดทุนหรือใช้ระยะเวลาในการนำเงินออกมาใช้สูงกว่าแต่สินทรัพย์ ประเภทนี้ จะก่อให้เกิดมูลค่าเพิ่มในอนาคตได้ เช่น กองทุนรวม หุ้น ที่ดิน

c) **สินทรัพย์ใช้ส่วนตัว** เป็นสินทรัพย์ที่มีสภาพคล่องต่ำที่สุดซึ่ง นำเงินออกมาใช้ได้ยากอาจจะต้องใช้ระยะเวลานานและเป็นสินทรัพย์ที่ใช้ส่วนตัวที่ใช้ในชีวิตประจำวันเช่น บ้าน รถยนต์

2) **หนี้สิ้น**

a) **หนี้สินระยะสั้น** เป็นหนี้สินที่ต้องใช้ชำระคืนในระยะสั้นและมักจะมีดอกเบี้ยที่สูง เช่น บัตรเครดิต

b) **หนี้สินระยะยาว** เป็นหนี้ที่ใช้ระยะเวลาในการชำระคืนนานและดอกเบี้ยต่ำกว่าหนี้สินระยะสั้น เช่น หนี้บ้าน หนี้รถเป็นต้น

แล้วแบ่งย่อยตามระยะเวลา

1) **สินทรัพย์**

a) **สินทรัพย์สภาพคล่อง** เป็นสินทรัพย์ที่มีสภาพคล่องสูงคือสามารถนำมาใช้จ่ายได้ง่าย อย่างเช่นเงินสด สามารถนำมาใช้จ่ายได้ทันที หรือ พวกกองทุนตลาดเงิน ซึ่งสามารถขายกองทุนแล้วนำเงินมาใช้โดยใช้ระยะ 1-2 วัน เพื่อกำหนดจัดงบประมาณสำหรับการใช้จ่ายโดยทั่วไป

b) **สินทรัพย์ลงทุน** เป็นสินทรัพย์ที่มีสภาพคล่องน้อยกว่าสินทรัพย์สภาพซึ่งถ้าขายมาใช้อาจจะเกิดการขาดทุนหรือใช้ระยะเวลาในการนำเงินออกมาใช้สูงกว่าแต่สินทรัพย์ ประเภทนี้ จะก่อให้เกิดมูลค่าเพิ่มในอนาคตได้ เช่น กองทุนรวม หุ้น ที่ดิน

c) **สินทรัพย์ใช้ส่วนตัว** เป็นสินทรัพย์ที่มีสภาพคล่องต่ำที่สุดซึ่ง นำเงินออกมาใช้ได้ยาก อาจจะต้องใช้ระยะเวลานานและเป็นสินทรัพย์ที่ใช้ส่วนตัวที่ใช้ในชีวิตประจำวันเช่น บ้าน รถยนต์

2) **หนี้สิน**

a) **หนี้สินระยะสั้น** เป็นหนี้สินที่ต้องใช้ชำระคืนในระยะสั้นและมักจะมีดอกเบี้ยที่สูง เช่น บัตรเครดิต

b) **หนี้สินระยะยาว** เป็นหนี้ที่ใช้ระยะเวลาในการชำระคืนนานและดอกเบี้ยต่ำกว่าหนี้สินระยะสั้น เช่น หนี้บ้าน หนี้รถเป็นต้น

ตาราง 2.1 ตัวอย่างงบดุล

สินทรัพย์		%	หนี้สินและความมั่งคั่ง		%
สินทรัพย์สภาพคล่อง			**หนี้สินระยะสั้น**		
เงินสด	5,000.00	**6.83**	ค่าสาธารณูปโภคค้างจ่าย	0.00	**0.00**
บัญชีเงินฝากออมทรัพย์	15,000.00	**20.52**	ยอดคงค้างหนี้สินบัตรเครดิต	0.00	**0.00**
บัญชีเงินฝากประจำ	40,000.00	**54.70**	หนี้สินเงินกู้ระยะสั้นอื่นๆ	0.00	**0.00**
ใบรับฝากเงินที่เปลี่ยนมือได้	0.00	**0.00**	อื่นๆ	0.00	**0.00**
อื่นๆ	0.00	**0.00**	**รวมหนี้สินระยะสั้น**	**0.00**	**0.00**
รวมสินทรัพย์สภาพคล่อง	60,000.00	82.05	**หนี้สินระยะยาว**		
สินทรัพย์เพื่อการลงทุน			ยอดคงค้างจากการกู้ยืมซื้อรถยนต์	0.00	**0.00**
พันธบัตร/หุ้นกู้	0.00	**0.00**	ยอดคงค้างจากการกู้ยืมซื้อบ้าน	0.00	**0.00**
หุ้นบุริมสิทธิ	0.00	**0.00**	หนี้สินระยะยาวอื่นๆ	0.00	**0.00**
หุ้นสามัญ	0.00	**0.00**	รวมหนี้สินระยะยาว	0.00	**0.00**
กองทุนรวม	0.00	**0.00**	รวมหนี้สิน	0.00	**0.00**
กองทุนสำรองเลี้ยงชีพ	3,118.65	**4.27**	**ความมั่งคั่งสุทธิ**	**73,118.65**	**100**
รวมสินทรัพย์เพื่อการลงทุน	**3,118.65**	**4.27**			
สินทรัพย์ใช้ส่วนตัวหรือสินทรัพย์มีค่า					
เครื่องประดับ	0.00	**0.00**			
รถจักรยานยนต์	10,000.00	**13.68**			
บ้าน	0.00	**0.00**			

ตาราง 2.1 ตัวอย่างงบดุล

สินทรัพย์		%	หนี้สินและความมั่งคั่ง		%
สินทรัพย์สภาพคล่อง			**หนี้สินระยะสั้น**		
เงินสด	5,000.00	6.83	ค่าสาธารณูปโภคค้างจ่าย	0.00	0.00
บัญชีเงินฝากออมทรัพย์	15,000.00	20.52	ยอดคงค้างหนี้สินบัตรเครดิต	0.00	0.00
บัญชีเงินฝากประจำ	40,000.00	54.70	หนี้สินเงินกู้ระยะสั้นอื่นๆ	0.00	0.00
ใบรับฝากเงินที่เปลี่ยนมือได้	0.00	0.00	อื่นๆ	0.00	0.00
อื่นๆ	0.00	0.00	**รวมหนี้สินระยะสั้น**	0.00	0.00
รวมสินทรัพย์สภาพคล่อง	60,000.00	82.05	**หนี้สินระยะยาว**		
สินทรัพย์เพื่อการลงทุน			ยอดคงค้างจากการกู้ยืมซื้อรถยนต์	0.00	0.00
พันธบัตร/หุ้นกู้	0.00	0.00	ยอดคงค้างจากการกู้ยืมซื้อบ้าน	0.00	0.00
หุ้นบุริมสิทธิ	0.00	0.00	หนี้สินระยะยาวอื่นๆ	0.00	0.00
หุ้นสามัญ	0.00	0.00	รวมหนี้สินระยะยาว	0.00	0.00
กองทุนรวม	0.00	0.00	รวมหนี้สิน	0.00	0.00
กองทุนสำรองเลี้ยงชีพ	3,118.65	4.27	**ความมั่งคั่งสุทธิ**	73,118.65	100
รวมสินทรัพย์เพื่อการลงทุน	3,118.65	4.27			
สินทรัพย์ใช้ส่วนตัวหรือสินทรัพย์มีค่า					
เครื่องประดับ	0.00	0.00			
รถจักรยานยนต์	10,000.00	13.68			
บ้าน	0.00	0.00			

2.1.5.2 งบกระแสเงินสด

งบกระแสเงินสดเป็นงบแสดงพฤติกรรมการใช้จ่ายของบุคคลนั้นๆ ทำให้แสดงถึงที่มาที่ไปในการใช้จ่าย

ตาราง 2.2 ตัวอย่างงบกระแสเงินสดส่วนบุคคล

กระแสเงินสดรับ		**%**
เงินเดือน (รวมค่าล่วงเวลา, ค่าคอมมิชชั่น, ค่าจ้าง, โบนัส)	177,000.00	100.00
รายได้อื่นๆ	0.00	0.00
กระแสเงินสดรับรวม	**177,000.00**	**100.00**
กระแสเงินสดจ่าย		
กระแสเงินสดจ่ายคงที่		
ค่าเบี้ยประกันชีวิต	13,593.00	7.68
ค่าประกันสังคม	6,276.00	3.55
เงินสะสมกองทุนสำรองเลี้ยงชีพ	4,680.00	2.64
กระแสเงินสดจ่ายคงที่รวม	24,549.00	13.87
กระแสเงินสดจ่ายผันแปร		
ค่าอาหาร	36,000.00	20.34
ค่าโทรศัพท์	4,800.00	2.71
ค่าสาธารณูปโภค (ค่าไฟฟ้า, ค่าน้ำประปา, อื่นๆ)	48,000.00	27.12
ค่าใช้จ่ายบันทนาการ	6,000.00	3.39
ค่าภาษีรถยนต์	500.00	0.28
ค่าใช้จ่ายในการเดินทาง	7,200.00	4.07
ค่าเสื้อผ้าและค่าใช้จ่ายในการบำรุงรักษาตนเองอื่นๆ	3,600.00	2.03
ค่าใช้จ่ายอื่นๆ	3,600.00	2.03
กระแสเงินสดจ่ายผันแปรรวม	109,700.00	61.97
กระแสเงินสดจ่ายเพื่อการออม / การลงทุน		
เงินออม	36,000.00	20.34
เงินลงทุน	0.00	0.00
กระแสเงินสดจ่ายเพื่อการออม / การลงทุนรวม	36,000.00	20.34
กระแสเงินสดจ่ายรวม	170,249.00	96.18
กระแสเงินสดสุทธิ	**6,751.00**	**3.82**

2.1.5.2 งบกระแสเงินสด

งบกระแสเงินสดเป็นงบแสดงพฤติกรรมการใช้จ่ายของบุคคลนั้นๆ ทำให้แสดงถึงที่มาที่ไปในการใช้จ่าย

ตาราง 2.2 ตัวอย่างงบกระแสเงินสดส่วนบุคคล

กระแสเงินสดรับ		**%**
เงินเดือน (รวมค่าล่วงเวลา, ค่าคอมมิชชั่น, ค่าจ้าง, โบนัส)	177,000.00	100.00
รายได้อื่นๆ	0.00	0.00
กระแสเงินสดรับรวม	177,000.00	100.00
กระแสเงินสดจ่าย		
กระแสเงินสดจ่ายคงที่		
ค่าเบี้ยประกันชีวิต	13,593.00	7.68
ค่าประกันสังคม	6,276.00	3.55
เงินสะสมกองทุนสำรองเลี้ยงชีพ	4,680.00	2.64
กระแสเงินสดจ่ายคงที่รวม	24,549.00	13.87
กระแสเงินสดจ่ายผันแปร		
ค่าอาหาร	36,000.00	20.34
ค่าโทรศัพท์	4,800.00	2.71
ค่าสาธารณูปโภค (ค่าไฟฟ้า, ค่าน้ำประปา, อื่นๆ)	48,000.00	27.12
ค่าใช้จ่ายบันทนาการ	6,000.00	3.39
ค่าภาษีรถยนต์	500.00	0.28
ค่าใช้จ่ายในการเดินทาง	7,200.00	4.07
ค่าเสื้อผ้าและค่าใช้จ่ายในการบำรุงรักษาตนเองอื่นๆ	3,600.00	2.03
ค่าใช้จ่ายอื่นๆ	3,600.00	2.03
กระแสเงินสดจ่ายผันแปรรวม	109,700.00	61.97
กระแสเงินสดจ่ายเพื่อการออม / การลงทุน		
เงินออม	36,000.00	20.34
เงินลงทุน	0.00	0.00
กระแสเงินสดจ่ายเพื่อการออม / การลงทุนรวม	36,000.00	20.34
กระแสเงินสดจ่ายรวม	170,249.00	96.18
กระแสเงินสดสุทธิ	6,751.00	3.82

งบกระแสเงินสดจะแบ่งออกเป็นใหญ่ๆ 2 ประเภทคือ รายรับและรายจ่าย

1) **รายรับ** คือเงินที่ได้เข้ามาซึ่งจะแตกต่างออกไปตามแต่ละบุคคล เช่น พนักงานบริษัท เจ้าของธรุกิจ เป็นต้น

2) **รายจ่าย** คือเงินที่ออกไปจะแบ่งย่อย ออกเป็นอีก 3 ประเภทด้วยกัน

a) ค่าใช้จ่ายในการลงทุน เป็นค่าใช้จ่ายสำหรับวางแผนในอนาคตตาเป้าหมายที่วางไว้

b) ค่าใช้จ่ายคงที่ เป็นค่าใช้จ่ายที่คงที่ เช่น ค่าผ่อนบ้าน ผ่อนรถเป็นต้น

c) ค่าใช้จ่ายเป็นค่าใช้จ่ายที่เปลี่ยนแปลงตลอด ขึ้นอยู่กับการใช้จ่ายของเราซึ่งเราสามารถกำหนดและเปลี่ยนแปลงในส่วนนี้ได้

2.1.6 การวิเคราะห์งบการเงินส่วนบุคคล

เมื่อได้งบการเงินมาเรียบร้อยแล้วเราสามารถนำตัวเลขในงบการเงินนั้นมาวิเคราะห์ได้โดยใช้อัตราส่วนการเงินต่างๆ ซึ่งเราได้อ้างอิงตัวเลขตามหลักการวางแผนการเงินส่วนบุคคล (CFP)

ตาราง 2.3 การคำนวณอัตราส่วนทางการเงินส่วนบุคคล

อัตราส่วน	สูตรการคำนวณ	ตัวเลขการคำนวณ	ผลลัพธ์
สภาพคล่อง (เท่า)	สินทรัพย์ที่มีสภาพคล่อง / หนี้สินระยะสั้น	60,000.00 / 0.00	อินฟินิตี้
สภาพคล่องพื้นฐาน (เดือน)	สินทรัพย์ที่มีสภาพคล่อง / กระแสเงินสดจ่ายต่อเดือน	60,000.00 / 11,187.42	5.36
สินทรัพย์ที่มีสภาพคล่องต่อความมั่งคั่งสุทธิ (%)	สินทรัพย์ที่มีสภาพคล่อง / ความมั่งคั่งสุทธิ	60,000.00 / 73,118.65	82.35
หนี้สินต่อสินทรัพย์ (%)	หนี้สินรวม / สินทรัพย์รวม	0.00 / 73,118.65	0.00
การชำระคืนหนี้จากรายได้ (%)	เงินชำระคืนหนี้สิน / รายรับรวม	0.00 / 177,000.00	0.00
การชำระคืนหนี้สินที่ไม่ใช่การจดจำนองจากรายได้ (%)	เงินชำระคืนหนี้สินไม่รวมภาระจดจำนอง / รายรับรวม	0.00 / 177,000.00	0.00
การออม (%)	เงินออม / รายรับรวม	36,000.00 / 177,000.00	20.33
การลงทุน (%)	สินทรัพย์ลงทุน / ความมั่งคั่งสุทธิ	3,118.65 / 73,118.65	4.26

งบกระแสเงินสดจะแบ่งออกเป็นใหญ่ๆ 2 ประเภทคือ รายรับและรายจ่าย

1) **รายรับ** คือเงินที่ได้เข้ามาซึ่งจะแตกต่างออกไปตามแต่ละบุคคล เช่น พนักงานบริษัท เจ้าของธรุกิจ เป็นต้น

2) **รายจ่าย** คือเงินที่ออกไปจะแบ่งย่อย ออกเป็นอีก 3 ประเภทด้วยกัน

a) ค่าใช้จ่ายในการลงทุน เป็นค่าใช้จ่ายสำหรับวางแผนในอนาคตตาเป้าหมายที่วางไว้

b) ค่าใช้จ่ายคงที่ เป็นค่าใช้จ่ายที่คงที่ เช่น ค่าผ่อนบ้าน ผ่อนรถเป็นต้น

c) ค่าใช้จ่ายเป็นค่าใช้จ่ายที่เปลี่ยนแปลงตลอด ขึ้นอยู่กับการใช้จ่ายของเราซึ่งเราสามารถกำหนดและเปลี่ยนแปลงในส่วนนี้ได้

2.1.6 การวิเคราะห์งบการเงินส่วนบุคคล

เมื่อได้งบการเงินมาเรียบร้อยแล้วเราสามารถนำตัวเลขในงบการเงินนั้นมาวิเคราะห์ได้โดยใช้อัตราส่วนการเงินต่างๆ ซึ่งเราได้อ้างอิงตัวเลขตามหลักการวางแผนการเงินส่วนบุคคล (CFP)

ตาราง 2.3 การคำนวณอัตราส่วนทางการเงินส่วนบุคคล

อัตราส่วน	สูตรการคำนวณ	ตัวเลขการคำนวณ	ผลลัพธ์
สภาพคล่อง (เท่า)	สินทรัพย์ที่มีสภาพคล่อง / หนี้สินระยะสั้น	60,000.00 / 0.00	อินฟินิตี้
สภาพคล่องพื้นฐาน (เดือน)	สินทรัพย์ที่มีสภาพคล่อง / กระแสเงินสดจ่ายต่อเดือน	60,000.00 / 11,187.42	5.36
สินทรัพย์ที่มีสภาพคล่องต่อความมั่งคั่งสุทธิ (%)	สินทรัพย์ที่มีสภาพคล่อง / ความมั่งคั่งสุทธิ	60,000.00 / 73,118.65	82.35
หนี้สินต่อสินทรัพย์ (%)	หนี้สินรวม / สินทรัพย์รวม	0.00 / 73,118.65	0.00
การชำระคืนหนี้จากรายได้ (%)	เงินชำระคืนหนี้สิน / รายรับรวม	0.00 / 177,000.00	0.00
การชำระคืนหนี้สินที่ไม่ใช่การจดจำนองจากรายได้ (%)	เงินชำระคืนหนี้สินไม่รวมภาระจดจำนอง / รายรับรวม	0.00 / 177,000.00	0.00
การออม (%)	เงินออม / รายรับรวม	36,000.00 / 177,000.00	20.33
การลงทุน (%)	สินทรัพย์ลงทุน / ความมั่งคั่งสุทธิ	3,118.65 / 73,118.65	4.26

2.1.6.1 การวิเคราะห์สภาพคล่อง

เป็นการวิเคราะห์ความสามารถในการจัดหาเงินสดเพื่อมาใช้จ่ายในชีวิตประจำวันและสำรองเผื่อฉุกเฉิน

a) **อัตราส่วนสภาพคล่อง** แสดงถึงความสามารถในการชำระหนี้ระยะสั้นว่ามีสินทรัพย์ที่มีสภาพคล่อง เช่น เงินสด หรือเงินฝากออมทรัพย์เพียงพอสำหรับชำระหนี้สินระยะสั้นที่จะครบกำหนดใน 1 ปีหรือไม่

b) **อัตราส่วนสภาพคล่องพื้นฐาน** แสดงถึงสินทรัพย์สภาพคล่องที่มีอยู่สามารถดำรงชีวิตได้กี่เดือน

c) **อัตราส่วนสินทรัพย์สภาพคล่องต่อความมั่งคั่งสุทธิ** แสดงถึงความมั่งคั่งสุทธิที่บุคคลมีอยู่นั้น มีสัดส่วนสินทรัพย์สภาพคล่องมากน้อยเพียงใด

2.1.6.2 การวิเคราะห์ภาระหนี้สิน

เป็นการวิเคราะห์ความเหมาะสมของระดับการก่อหนี้สินของบุคคลในปัจจุบันและความสามารถในการชำระหนี้สินในอนาคตได้

a) **อัตราส่วนหนี้สินต่อสินทรัพย์** แสดงถึงระดับหนี้สินที่บุคคลมีภาระผูกพันต้องชำระคืนในอนาคตว่ามีมากน้อยเพียงใด ซึ่งให้ทราบว่าบุคคลนั้นได้มาจากการจัดหาเงินทุนจากหนี้สินจำนวนเท่าใด

b) **อัตราส่วนแสดงการชำระคืนหนี้สินจากรายได้** แสดงถึงความสามารถในการชำระหนี้สินจากรายได้รวมที่ได้รับ

c) **อัตราส่วนแสดงการชำระคืนหนี้สินที่ไม่ใช่การจดจำนองจากรายได้** ในการชำระหนี้สินที่ไม่ใช่การจดจำนองจากรายได้รวม ภาระหนี้สินที่ใช้ในการคำนวณ คือ ภาระหนี้สินทุกชนิดยกเว้นเกิดจากการจดจำนอง เช่น บ้านและที่ดิน

2.1.6.3 อัตราส่วนการออมและการลงทุน

เป็นการวิเคราะห์ความเหมาะสมในการออมและลงทุนของบุคคลในปัจจุบันว่ามีความสอดคล้องกับเป้าหมายทางการเงินทั้งในระยะสั้นและระยะยาวหรือไม่

2.1.6.1 การวิเคราะห์สภาพคล่อง

เป็นการวิเคราะห์ความสามารถในการจัดหาเงินสดเพื่อมาใช้จ่ายในชีวิตประจำวันและสำรองเผื่อฉุกเฉิน

a) อัตราส่วนสภาพคล่อง แสดงถึงความสามารถในการชำระหนี้ระยะสั้นว่ามีสินทรัพย์ที่มีสภาพคล่อง เช่น เงินสด หรือเงินฝากออมทรัพย์เพียงพอสำหรับชำระหนี้สินระยะสั้นที่จะครบกำหนดใน 1 ปีหรือไม่

b) อัตราส่วนสภาพคล่องพื้นฐาน แสดงถึงสินทรัพย์สภาพคล่องที่มีอยู่สามารถดำรงชีวิตได้กี่เดือน

c) อัตราส่วนสินทรัพย์สภาพคล่องต่อความมั่งคั่งสุทธิ แสดงถึงความมั่งคั่งสุทธิที่บุคคลมีอยู่นั้น มีสัดส่วนสินทรัพย์สภาพคล่องมากน้อยเพียงใด

2.1.6.2 การวิเคราะห์ภาระหนี้สิน

เป็นการวิเคราะห์ความเหมาะสมของระดับการก่อหนี้สินของบุคคลในปัจจุบันและความสามารถในการชำระหนี้สินในอนาคตได้

a) อัตราส่วนหนี้สินต่อสินทรัพย์ แสดงถึงระดับหนี้สินที่บุคคลมีภาระผูกพันต้องชำระคืนในอนาคตว่ามีมากน้อยเพียงใด ซึ่งให้ทราบว่าบุคคลนั้นได้มาจากการจัดหาเงินทุนจากหนี้สินจำนวนเท่าใด

b) อัตราส่วนแสดงการชำระคืนหนี้สินจากรายได้ แสดงถึงความสามารถในการชำระหนี้สินจากรายได้รวมที่ได้รับ

c) อัตราส่วนแสดงการชำระคืนหนี้สินที่ไม่ใช่การจดจำนองจากรายได้ ในการชำระหนี้สินที่ไม่ใช่การจดจำนองจากรายได้รวม ภาระหนี้สินที่ใช้ในการคำนวณ คือ ภาระหนี้สินทุกชนิดยกเว้นเกิดจากการจดจำนอง เช่น บ้านและที่ดิน

2.1.6.3 อัตราส่วนการออมและการลงทุน

เป็นการวิเคราะห์ความเหมาะสมในการออมและลงทุนของบุคคลในปัจจุบันว่ามีความสอดคล้องกับเป้าหมายทางการเงินทั้งในระยะสั้นและระยะยาวหรือไม่

a) อัตราส่วนการออม แสดงถึงสัดส่วนของเงินออมที่บุคคลกันไว้จากรายได้รวม เพื่อการอุปโภค บริโภค หรือใช้จ่ายในอนาคต รวมถึงเป็นเงินสำรองที่ใช้ในยามเกษียณอายุ

b) อัตราส่วนการลงทุน แสดงถึงสินทรัพย์ที่บุคคลลงทุนไว้โดยมุ่งหวังให้ได้ผลตอบแทนและเป็นแหล่งที่มาของรายได้เพิ่มขึ้นและเพื่อบรรลุเป้าหมายอิสรภาพการเงินรวมทั้งเป้าหมายอื่นๆ

ตาราง 2.4 การเปรียบเทียบผลการคำนวณอัตราส่วนแต่ละด้านกับเกณฑ์มาตรฐาน

อัตราส่วน	เกณฑ์มาตรฐาน	ผลการคำนวณ
การวิเคราะห์ด้านสภาพคล่อง		
อัตราส่วนสภาพคล่อง	> 1	อินฟินิตี้ เท่า
อัตราส่วนสภาพคล่องพื้นฐาน	3 – 6 เดือน	5.36 เดือน
อัตราส่วนสินทรัพย์ที่มีสภาพคล่องต่อความมั่งคั่งสุทธิ	≥15%	82.35 %
การวิเคราะห์ด้านหนี้สิน		
อัตราส่วนหนี้สินต่อสินทรัพย์	< 50%	0.00 %
อัตราส่วนแสดงการชำระคืนหนี้สินจากรายได้	< 35 – 45 %	0.00 %
อัตราส่วนการชำระคืนหนี้สินที่ไม่ใช่การจดจำนองจากรายได้	< 15 – 20%	0.00 %
การวิเคราะห์ด้านการออมและการลงทุน		
อัตราส่วนการออม	> 10%	20.33 %
อัตราส่วนการลงทุน	≥ 50%	4.26 %

2.1.7 ภาษีบุคคลธรรมดา

ในการวางแผนงบการเงินส่วนบุคคลต้องมีการวางแผนทางภาษีร่วมด้วย โดยอัตราภาษีตามขั้นบันไดและค่าลดหย่อน

a) **อัตราส่วนการออม** แสดงถึงสัดส่วนของเงินออมที่บุคคลกันไว้จากรายได้รวม เพื่อการอุปโภค บริโภค หรือใช้จ่ายในอนาคต รวมถึงเป็นเงินสำรองที่ใช้ในยามเกษียณอายุ

b) **อัตราส่วนการลงทุน** แสดงถึงสินทรัพย์ที่บุคคลลงทุนไว้โดยมุ่งหวังให้ได้ผลตอบแทนและเป็นแหล่งที่มาของรายได้เพิ่มขึ้นและเพื่อบรรลุเป้าหมายอิสรภาพการเงินรวมทั้งเป้าหมายอื่นๆ

ตาราง 2.4 การเปรียบเทียบผลการคำนวณอัตราส่วนแต่ละด้านกับเกณฑ์มาตรฐาน

อัตราส่วน	เกณฑ์มาตรฐาน	ผลการคำนวณ
การวิเคราะห์ด้านสภาพคล่อง		
อัตราส่วนสภาพคล่อง	> 1	อินฟินิตี้ เท่า
อัตราส่วนสภาพคล่องพื้นฐาน	3 – 6 เดือน	5.36 เดือน
อัตราส่วนสินทรัพย์ที่มีสภาพคล่องต่อความมั่งคั่งสุทธิ	≥15%	82.35 %
การวิเคราะห์ด้านหนี้สิน		
อัตราส่วนหนี้สินต่อสินทรัพย์	< 50%	0.00 %
อัตราส่วนแสดงการชำระคืนหนี้สินจากรายได้	< 35 – 45 %	0.00 %
อัตราส่วนการชำระคืนหนี้สินที่ไม่ใช่การจดจำนองจากรายได้	< 15 – 20%	0.00 %
การวิเคราะห์ด้านการออมและการลงทุน		
อัตราส่วนการออม	> 10%	20.33 %
อัตราส่วนการลงทุน	≥ 50%	4.26 %

2.1.7 ภาษีบุคคลธรรมดา

ในการวางแผนงบการเงินส่วนบุคคลต้องมีการวางแผนทางภาษีร่วมด้วย โดยอัตราภาษีตามขั้นบันไดและค่าลดหย่อน

ตาราง 2.5 ตารางอัตราเสียภาษีบุคคลธรรมดา

เงินได้สุทธิ	ช่วงเงินได้สุทธิ แต่ละขั้น	อัตราภาษี ร้อยละ
1 - 150,000	150,000	ได้รับยกเว้น
150,001 - 300,000	150,000	5
300,001 - 500,000	200,000	10
500,001 - 750,000	250,000	15
750,001 - 1,000,000	250,000	20
1,000,001 - 2,000,000	1,000,000	25
2,000,001 - 4,000,000	2,000,000	30
4,000,001 บาทขึ้นไป	-	35

ตาราง 2.6 ตารางค่าลดหย่อนต่างๆ ของภาษีบุคคลธรรมดา

รายการค่าลดหย่อน	โสด	สามี - ภรรยา	
		แยกยื่น	ยื่นแบบรวมกัน
1. ส่วนตัว	30,000	30,000	60,000
2. บุตร (คน)	-	15,000	30,000
3. การศึกษาบุตร (คน)	-	2,000	4,000
4. ดอกเบี้ยเงินกู้ยืม	-	15,000	30,000
- ต่างฝ่ายต่างกู้ยืม	100,000	100,000	200,000
- กู้ร่วมกัน	กึ่งหนึ่งรวมกันไม่เกิน 100,000	กึ่งหนึ่งรวมกันไม่เกิน 100,000	100,000
5. เบี้ยประกันชีวิต	100,000	100,000	200,000
6. เบี้ยประกันชีวิต (บำนาญ)/คน	15% ของเงินได้ สูงสุด 200,000	15% ของเงินได้ สูงสุด 200,000	15% ของเงินได้ สูงสุด (รวมกัน) 400,000
7. กองทุนสำรองเลี้ยงชีพ	500,000	500,000	1,000,000
8. RMF	15% ของเงินได้ สูงสุด 500,000	15% ของเงินได้ สูงสุด 500,000	15% ของเงินได้ สูงสุด (รวมกัน) 1,000,000
9. LTF	15% ของเงินได้ สูงสุด 500,000	15% ของเงินได้ สูงสุด 500,000	15% ของเงินได้ สูงสุด (รวมกัน) 1,000,000
10. กองทุนประกันสังคม	ตามที่จ่ายจริง	ตามที่จ่ายจริง	ตามที่จ่ายจริง
11. ค่าอุปการะบิดามารดา (คน)	30,000	30,000	60,000
12. ค่าอุปการะเลี้ยงดูคนพิการ	60,000	60,000	120,000

ตาราง 2.5 ตารางอัตราเสียภาษีบุคคลธรรมดา

เงินได้สุทธิ	ช่วงเงินได้สุทธิ แต่ละขั้น	อัตราภาษี ร้อยละ
1 - 150,000	150,000	ได้รับยกเว้น
150,001 - 300,000	150,000	5
300,001 - 500,000	200,000	10
500,001 - 750,000	250,000	15
750,001 - 1,000,000	250,000	20
1,000,001 - 2,000,000	1,000,000	25
2,000,001 - 4,000,000	2,000,000	30
4,000,001 บาทขึ้นไป	-	35

ตาราง 2.6 ตารางค่าลดหย่อนต่างๆ ของภาษีบุคคลธรรมดา

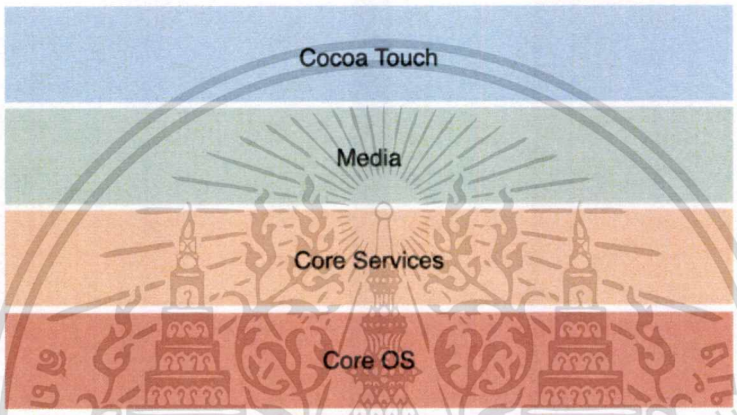
รายการค่าลดหย่อน	โสด	สามี – ภรรยา	
		แยกยื่น	ยื่นแบบรวมกัน
1. ส่วนตัว	30,000	30,000	60,000
2. บุตร (คน)	-	15,000	30,000
3. การศึกษาบุตร (คน)	-	2,000	4,000
4. ดอกเบี้ยเงินกู้ยืม	-	15,000	30,000
- ต่างฝ่ายต่างกู้ยืม	100,000	100,000	200,000
- กู้ร่วมกัน	ที่งหนึ่งรวมกันไม่เกิน 100,000	ที่งหนึ่งรวมกันไม่เกิน 100,000	100,000
5. เบี้ยประกันชีวิต	100,000	100,000	200,000
	15% ของเงินได้ สูงสุด	15% ของเงินได้ สูงสุด	15% ของเงินได้ สูงสุด
6. เบี้ยประกันชีวิต (บำนาญ)/คน	200,000	200,000	(รวมกัน) 400,000
7. กองทุนสำรองเลี้ยงชีพ	500,000	500,000	1,000,000
	15% ของเงินได้ สูงสุด	15% ของเงินได้ สูงสุด	15% ของเงินได้ สูงสุด
8. RMF	500,000	500,000	(รวมกัน) 1,000,000
	15% ของเงินได้ สูงสุด	15% ของเงินได้ สูงสุด	15% ของเงินได้ สูงสุด
9. LTF	500,000	500,000	(รวมกัน) 1,000,000
10. กองทุนประกันสังคม	ตามที่จ่ายจริง	ตามที่จ่ายจริง	ตามที่จ่ายจริง
11. ค่าอุปการะบิดามารดา (คน)	30,000	30,000	60,000
12. ค่าอุปการะเลี้ยงดูคนพิการ	60,000	60,000	120,000

2.2 ทฤษฎีเบื้องต้นเกี่ยวกับระบบ iOS

ไอโอเอส (ก่อนหน้านี้ใช้ชื่อ **ไอโฟนโอเอส**) คือระบบปฏิบัติการบนอุปกรณ์พกพาพัฒนาและจำหน่ายโดยแอปเปิล (บริษัท) เปิดตัวครั้งแรกในปี 2007 เพื่อใช้บนไอโฟน และได้มีการพัฒนาเพิ่มเติมเพื่อใช้บนอุปกรณ์พกพา ของแอปเปิล

2.2.1 สถาปัตยกรรมของระบบ iOS

ในการเขียนโปรแกรมนั้น ถ้าเราเขียนใน เลเวลที่สูงกว่าจะสามารถทำได้ง่ายและสูงกว่าการเขียนโปรแกรมในเลเวลที่ต่ำกว่าและยังสามารถลดจำนวนโค้ดในการเขียนโปรแกรมอีกด้วย



รูป 2.3 แสดงระดับชั้นของระบบ iOS

2.2.2 โครงสร้างไฟล์พื้นฐานของ iOS

โครงสร้างไฟล์ที่ XCode สร้างขึ้นมาเพื่อรองรับการทำงานของ iOS Application ดังภาพ

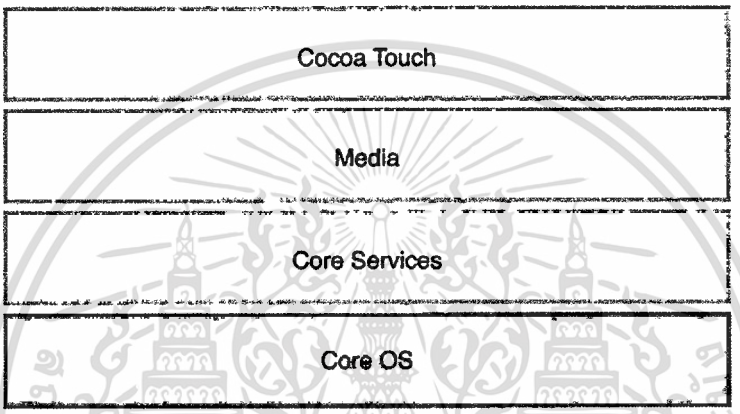
รูป 2.4 แสดงไฟล์พื้นฐานของ iOS

2.2 ทฤษฎีเบื้องต้นเกี่ยวกับระบบ iOS

ไอโอเอส (ก่อนหน้านี้ใช้ชื่อ **ไอโฟนโอเอส**) คือระบบปฏิบัติการบนอุปกรณ์พกพาพัฒนาและจำหน่ายโดยแอปเปิล (บริษัท) เปิดตัวครั้งแรกในปี 2007 เพื่อใช้บนไอโฟน และได้มีการพัฒนาเพิ่มเติมเพื่อใช้บนอุปกรณ์พกพา ของแอปเปิล

2.2.1 สถาปัตยกรรมของระบบ iOS

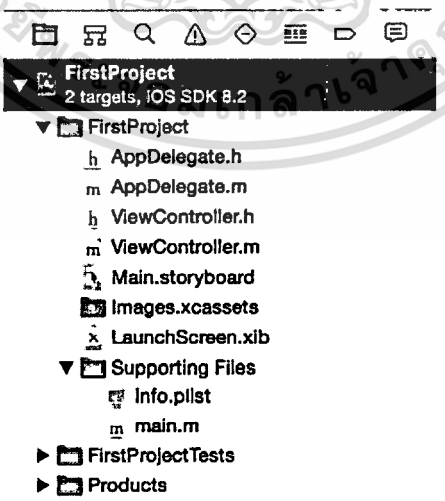
ในการเขียนโปรแกรมนั้น ถ้าเราเขียนใน เลเวลที่สูงกว่าจะสามารถทำได้ง่ายและสูงกว่าการเขียนโปรแกรมในเลเวลที่ต่ำกว่าและยังสามารถลดจำนวนโค้ดในการเขียนโปรแกรมอีกด้วย



รูป 2.3 แสดงระดับชั้นของระบบ iOS

2.2.2 โครงสร้างไฟล์พื้นฐานของ iOS

โครงสร้างไฟล์ที่ XCode สร้างขึ้นมาเพื่อรองรับการทำงานของ iOS Application ดังภาพ



รูป 2.4 แสดงไฟล์พื้นฐานของ iOS

จากภาพด้านบน เราจะเห็นว่ามีไฟล์ต่าง ๆ ดังนี้

1) **AppDelegate** มีหน้าที่ในการควบคุมการทำงานต่าง ๆ ของ Application ไม่ว่า จะเป็นการควบคุมสถานะของแอพพลิเคชัน (Application Life Cycle) หรือแม้กระทั่งการรับข้อมูล Push Notification นั่นเอง
2) **ViewController** เป็นไฟล์ Controller สำหรับควบคุมการทำงานและการแสดงผลของหน้าจอบน Application ซึ่งไฟล์ ViewController ต่าง ๆ จะถูกเรียกใช้ก็ต่อเมื่อเราเชื่อมต่อมันเข้ากับ View บน Storyboard โดยเราสามารถสร้าง ViewController จำนวนเท่าไรก็ได้ขึ้นอยู่กับหน้าจอการแสดงผลของ Application
3) **Main.storyboard** สำหรับวาง layout หน้าจอโปรแกรมของเรา เป็นไฟล์ที่รวม View หลาย ๆ View ไว้
4) **Images.xcassets** เป็นไฟล์ที่ช่วยให้เราจัดการ Application Icon ขนาดต่าง ๆ และ Splashscreen (ในเวอร์ชันก่อน iOS 8.0 จะต้องใส่ภาพ Splashscreen เข้าไปด้วย - Splashscreen คือหน้าจอที่แสดงผลหลังจากกดเปิด Application ซึ่งจะแสดงเพียงแค่ 1-3 วินาทีเท่านั้น)
5) **LaunchScreen.xib** หลังจาก Apple ออก iOS8.0 ได้เปลี่ยนแปลงวิธีการจัดการ Splashscreen จากเดิมอยู่ที่ Images.xcassets ให้มาจัดการที่ไฟล์ LaunchScreen.xib แทน โดยข้อดีของมันคือ เราสามารถใส่ Object Component ต่าง ๆ เข้าไปได้เช่น TextField, ImageView เป็นต้น ซึ่งเวอร์ชันก่อน iOS 8.0 ที่จัดการผ่าน Images.xcassets จะไม่สามารถใส่ Object Component ใด ๆ เข้าไปได้ สามารถใส่ได้เพียงรูปภาพเท่านั้น
6) **Info.plist** เป็นไฟล์ Config ค่าต่าง ๆ เกี่ยวกับ Application เช่น Version, Application Name, Bundle Identifier เป็นต้น

2.3 ทฤษฎีเบื้องต้นเกี่ยวกับภาษา Swift

Swift เป็นภาษาใหม่สำหรับ iOS, OS X และ watchOS ที่สร้างจากภาษา C และ Objective-C ซึ่งพัฒนาให้มีคุณสมบัติที่ทันสมัยที่จะทำให้การเขียนโปรแกรมได้ง่ายขึ้นและมีความยืดหยุ่นมากขึ้น

Swift เป็นภาษาที่ออกแบบให้มีประสิทธิภาพสูงและง่ายต่อการพัฒนาโดยนำข้อดีของภาษาสมัยใหม่เข้ามามากมาย เช่น Type Inference, Clean Syntax, No semicolons, Closures, Generics ซึ่งคุณสมบัติที่กล่าวมาบางอย่างก็มีอยู่แล้วในภาษา Objective-C แต่ใน Swift นั้นจะพัฒนาให้น่าใช้มากยิ่งขึ้นและภาษา Swift ยังถูกออกแบบให้มีความปลอดภัยในการเขียน โปรแกรมมากขึ้น ทำให้ลดข้อผิดพลาดของโปรแกรมที่พัฒนาขึ้น ยกตัวอย่างเช่น

จากภาพด้านบน เราจะเห็นว่ามีไฟล์ต่าง ๆ ดังนี้

1) **AppDelegate** มีหน้าที่ในการควบคุมการทำงานต่าง ๆ ของ Application ไม่ว่า จะเป็นการควบคุมสถานะของแอพพลิเคชัน (Application Life Cycle) หรือแม้กระทั่งการรับข้อมูล Push Notification นั่นเอง

2) **ViewController** เป็นไฟล์ Controller สำหรับควบคุมการทำงานและการแสดงผลของหน้าจอบน Application ซึ่งไฟล์ ViewController ต่าง ๆ จะถูกเรียกใช้ก็ต่อเมื่อเราเชื่อมต่อมันเข้ากับ View บน Storyboard โดยเราสามารถสร้าง ViewController จำนวนเท่าไรก็ได้ขึ้นอยู่กับหน้าจอการแสดงผลของ Application

3) **Main.storyboard** สำหรับวาง layout หน้าจอโปรแกรมของเรา เป็นไฟล์ที่รวม View หลาย ๆ View ไว้

4) **Images.xcassets** เป็นไฟล์ที่ช่วยให้เราจัดการ Application Icon ขนาดต่าง ๆ และ Splashscreen (ในเวอร์ชันก่อน iOS 8.0 จะต้องใส่ภาพ Splashscreen เข้าไปด้วย - Splashscreen คือหน้าจอที่แสดงผลหลังจากกดเปิด Application ซึ่งจะแสดงเพียงแค่ 1-3 วินาทีเท่านั้น)

5) **LaunchScreen.xib** หลังจาก Apple ออก iOS8.0 ได้เปลี่ยนแปลงวิธีการจัดการ Splashscreen จากเดิมอยู่ที่ Images.xcassets ให้มาจัดการที่ไฟล์ LaunchScreen.xib แทน โดยข้อดีของมันคือ เราสามารถใส่ Object Component ต่าง ๆ เข้าไปได้เช่น TextField, ImageView เป็นต้น ซึ่งเวอร์ชันก่อน iOS 8.0 ที่จัดการผ่าน Images.xcassets จะไม่สามารถใส่ Object Component ใด ๆ เข้าไปได้ สามารถใส่ได้เพียงรูปภาพเท่านั้น

6) **Info.plist** เป็นไฟล์ Config ค่าต่าง ๆ เกี่ยวกับ Application เช่น Version, Application Name, Bundle Identifier เป็นต้น

2.3 ทฤษฎีเบื้องต้นเกี่ยวกับภาษา Swift

Swift เป็นภาษาใหม่สำหรับ iOS, OS X และ watchOS ที่สร้างจากภาษา C และ Objective-C ซึ่งพัฒนาให้มีคุณสมบัติที่ทันสมัยที่จะทำให้การเขียนโปรแกรมได้ง่ายขึ้นและมีความยืดหยุ่นมากขึ้น

Swift เป็นภาษาที่ออกแบบให้มีประสิทธิภาพสูงและง่ายต่อการพัฒนาโดยนำข้อดีของภาษาสมัยใหม่เข้ามามากมาย เช่น Type Inference, Clean Syntax, No semicolons, Closures, Generics ซึ่งคุณสมบติที่กล่าวมาบางอย่างก็มีอยู่แล้วในภาษา Objective-C แต่ใน Swift นั้นจะพัฒนาให้น่าใช้มากยิ่งขึ้นและภาษา Swift ยังถูกออกแบบให้มีความปลอดภัยในการเขียน โปรแกรมมากขึ้น ทำให้ลดข้อผิดพลาดของโปรแกรมที่พัฒนาขึ้น ยกตัวอย่างเช่น

1) ไม่อนุญาติให้มีตัวแปรที่ไม่ได้ถูกกำหนดค่าในโปรแกรม

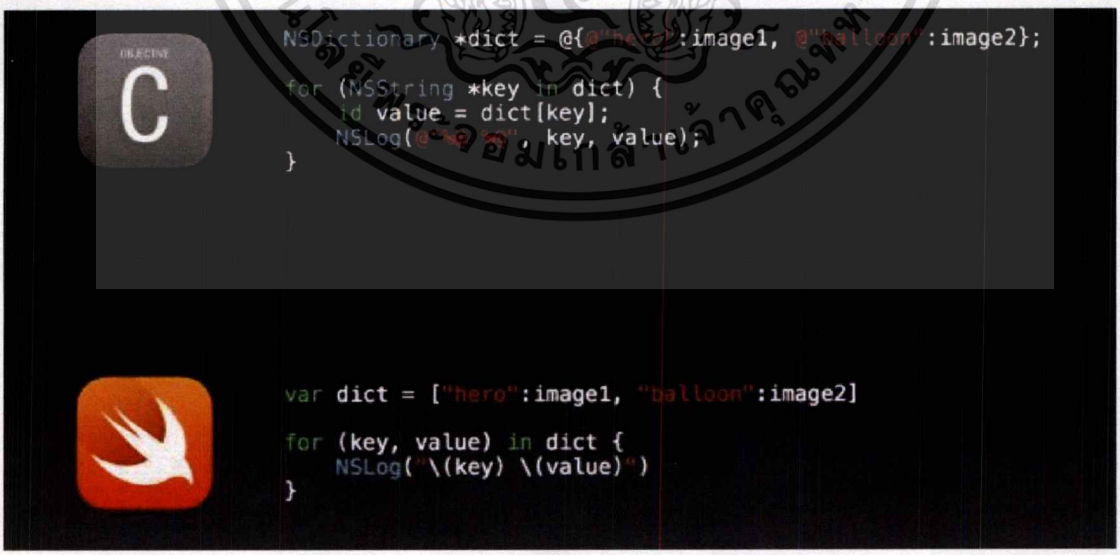
2) ไม่ต้องเขียนสัญลักษณ์ * (Asterisk) ขณะประกาศตัวแปร Pointer

3) ตรวจสอบการใช้งานค่าต่ำสุดและสูงสุดของตัวเลขจำนวนเต็ม

4) จะต้องเขียนวงเล็บปีกกาครอบส่วนของโปรแกรมที่อยู่ภายใต้เงื่อนไขใดๆ

2.3.1 ข้อแตกต่างระหว่างภาษา Swift และ Objective-C

รูป 2.5 Dictionary Creation and How to Access



รูป 2.6 String Sorting

1) ไม่อนุญาติให้มีตัวแปรที่ไม่ได้ถูกกำหนดค่าในโปรแกรม

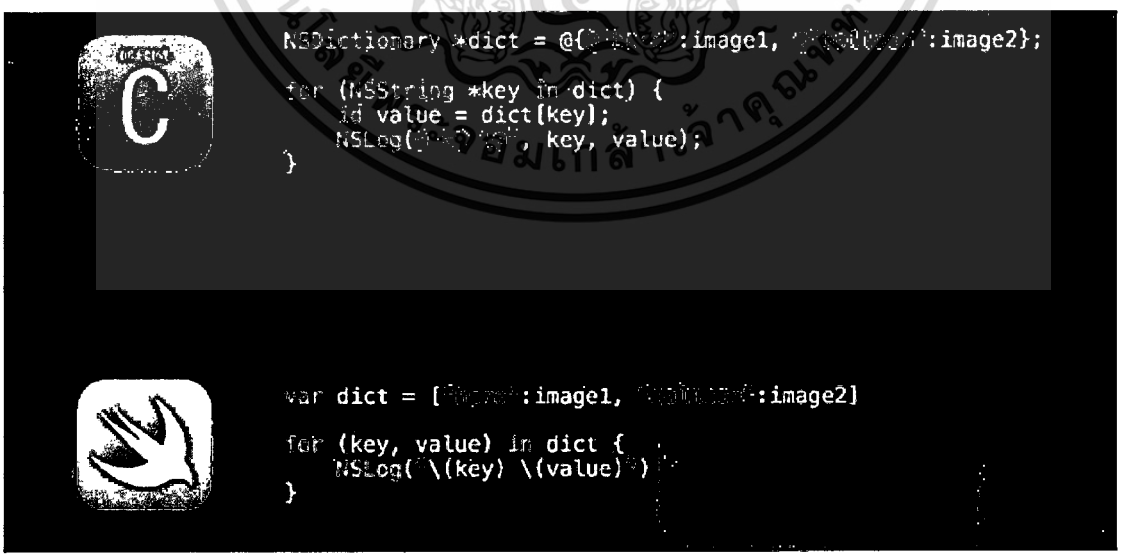
2) ไม่ต้องเขียนสัญลักษณ์ * (Asterisk) ขณะประกาศตัวแปร Pointer

3) ตรวจสอบการใช้งานค่าต่ำสุดและสูงสุดของตัวเลขจำนวนเต็ม

4) จะต้องเขียนวงเล็บปีกกาครอบส่วนของโปรแกรมที่อยู่ภายใต้เงื่อนไขใดๆ

2.3.1 ข้อแตกต่างระหว่างภาษา Swift และ Objective-C

รูป 2.5 Dictionary Creation and How to Access



รูป 2.6 String Sorting

2.3.2 อนาคตของ Swift และการเขียนแอพบน iOS

1) Apple ต้องการให้ Swift เป็นภาษาที่จะมา **"แทนที่"** Objective-C แทนที่จะเป็น **"ทางเลือก"**

2) คิดว่ายังคงต้องใช้เวลาเปลี่ยนผ่านและการพัฒนาสักระยะใหญ่ๆ ทั้งเรื่องของความสามารถของภาษาและ Community

3) Objective-C เป็นภาษาเก่าแก่ที่ถูกพัฒนามากว่า 20 ปี คาดว่าหลายต่อหลายคนยังคงจะใช้ Objective-C ไปก่อนอีกนาน

4) Apple ต้องพยายามลดช่องว่างความยากของตัวภาษาในการเรียนรู้การพัฒนาโปรแกรมบน Mac OS X และ iOS เพื่อจูงใจนักพัฒนาหน้าใหม่ๆหลายคนที่ปรับตัวเองมาจากภาษาสคริปต์สมัยใหม่อย่าง Python, Ruby, Javascript ก็น่าจะเป็นอีกกลุ่มที่สนใจใช้ Swift ในการพัฒนา

2.3.3 โครงสร้างภาษา Swift

การประกาศตัวแปร

การประกาศตัวแปรในภาษา Swift นั้นประกาศได้ 2 แบบ คือ

1) แบบระบุชนิดตัวแปรเข้าไปเลย (Type Safe)

2) แบบไม่ระบุชนิดตัวแปร (Type Inference)

การประกาศแบบ Type Safe จะทำให้เรารู้ได้เลยว่าชนิดข้อมูลของตัวแปรนั้นๆ เป็นชนิดอะไร จะทำให้ปลอดภัยจากการใช้งานสลับชนิดตัวแปรด้วย แต่การประกาศแบบ Type Inference เป็นการประกาศที่พบบ่อยมาก เพราะมันทำให้เราเขียนโค้ดได้สั้นลง ส่วนการรู้ว่าเป็นชนิดอะไรนั้นเราต้องไปดูที่ค่าของตัวแปรเอาเองว่าเป็นชนิดอะไร

การตัวแปรในภาษา Swift มีอยู่ด้วยกัน 2 ประเภทหลัก คือ

1) ตัวแปรที่ไม่สามารถเปลี่ยนแปลงค่าได้ หรือ Constant ตัวแปรนี้จะประกาศตัวแปรด้วยคีย์เวิร์ด **let**

2) ตัวแปรที่สามารถเปลี่ยนแปลงค่าหลังจากประกาศตัวแปรแล้วได้ หรือ Variable ตัวแปรนี้จะประกาศตัวแปรด้วยคีย์เวิร์ด **var**

2.3.2 อนาคตของ Swift และการเขียนแอพบน iOS

1) Apple ต้องการให้ Swift เป็นภาษาที่จะมา **"แทนที่"** Objective-C แทนที่จะเป็น **"ทางเลือก"**

2) คิดว่ายังคงต้องใช้เวลาเปลี่ยนผ่านและการพัฒนาสักระยะใหญ่ๆ ทั้งเรื่องของความสามารถของภาษาและ Community

3) Objective-C เป็นภาษาเก่าแก่ที่ถูกพัฒนามากว่า 20 ปี คาดว่าหลายต่อหลายคนยังคงจะใช้ Objective-C ไปก่อนอีกนาน

4) Apple ต้องพยายามลดช่องว่างความยากของตัวภาษาในการเรียนรู้การพัฒนาโปรแกรมบน Mac OS X และ iOS เพื่อจูงใจนักพัฒนาหน้าใหม่ๆหลายคนที่ปรับตัวเองมาจากภาษาสคริปต์สมัยใหม่อย่าง Python, Ruby, Javascript ก็น่าจะเป็นอีกกลุ่มที่สนใจใช้ Swift ในการพัฒนา

2.3.3 โครงสร้างภาษา Swift

การประกาศตัวแปร

การประกาศตัวแปรในภาษา Swift นั้นประกาศได้ 2 แบบ คือ

1) แบบระบุชนิดตัวแปรเข้าไปเลย (Type Safe)

2) แบบไม่ระบุชนิดตัวแปร (Type Inference)

การประกาศแบบ Type Safe จะทำให้เรารู้ได้เลยว่าชนิดข้อมูลของตัวแปรนั้นๆ เป็นชนิดอะไร จะทำให้ปลอดภัยจากการใช้งานสลับชนิดตัวแปรด้วย แต่การประกาศแบบ Type Inference เป็นการประกาศที่พบบ่อยมาก เพราะมันทำให้เราเขียนโค้ดได้สั้นลง ส่วนการรู้ว่าเป็นชนิดอะไรนั้นเราต้องไปดูที่ค่าของตัวแปรเอาเองว่าเป็นชนิดอะไร

การตัวแปรในภาษา Swift มีอยู่ด้วยกัน 2 ประเภทหลัก คือ

1) ตัวแปรที่ไม่สามารถเปลี่ยนแปลงค่าได้ หรือ Constant ตัวแปรนี้จะประกาศตัวแปรด้วยคีย์เวิร์ด **let**

2) ตัวแปรที่สามารถเปลี่ยนแปลงค่าหลังจากประกาศตัวแปรแล้วได้ หรือ Variable ตัวแปรนี้จะประกาศตัวแปรด้วยคีย์เวิร์ด **var**

ตัวอย่าง 2.1 การประกาศตัวแปร

```
/แสดงข้อความธรรมดา
println("Hello Let Swift.")
//แสดงค่าตัวแปรแบบไม่มีข้อความ
println(langName)
//แสดงค่าตัวแปรกับข้อความ
println("Hi! \(langName)")
```

การปริ้นและแสดงข้อความ

ใช้คำสั่ง println() หรือไม่ก็ print() เพื่อปริ้นค่าออกมาตามนี้

ตัวอย่าง 2.2 การปริ้นข้อความ

```
//ตัวแปร
var tools = 4
//println("tools number \(tools)")


//constants and variables
//type safe
var langaugeName: String = "Swift"
var version: Double = 1.0
let introduced: Int = 2012
let isAwesome: Bool = true //use true or false

//type inference
var langName = "Swift inf"  //inferred as String
var version2 = 2.0          //inferred as Double
let intro = 2014            //inferred as Int
let awesome = true          //inferred as Bool
```

2.3.4 Realm Object

Realm Object เหมือนกับ object ธรรมดา มีคุณสมบัติแต่ว่าต้องเป็น dynamic มีการประกาศตัวแปรเป็นเชิงคลาส

ตัวอย่าง 2.1 การประกาศตัวแปร

```
/แสดงข้อความธรรมดา
println("Hello Let Swift.")
//แสดงค่าตัวแปรแบบไม่มีข้อความ
println(langName)
//แสดงค่าตัวแปรกับข้อความ
println("Hi! \(langName)")
```

การปริ้นและแสดงข้อความ

ใช้คำสั่ง println() หรือไม่ก็ print() เพื่อปริ้นค่าออกมาตามนี้

ตัวอย่าง 2.2 การปริ้นข้อความ

```
//ตัวแปร
var tools = 4
//println("tools number \(tools)")


//constants and variables
//type safe
var langaugeName: String = "Swift"
var version: Double = 1.0
let introduced: Int = 2012
let isAwesome: Bool = true //use true or false

//type inference
var langName = "Swift inf"  //inferred as String
var version2 = 2.0          //inferred as Double
let intro = 2014            //inferred as Int
let awesome = true          //inferred as Bool
```

2.3.4 Realm Object

Realm Object เหมือนกับ object ธรรมดา มีคุณสมบัติแต่ว่าต้องเป็น dynamic มีการประกาศตัวแปรเป็นเชิงคลาส

โปรแกรม 2.3 Realm Object

```
// Realm Object
class Dog: Object {
   dynamic var name = ""
   dynamic var age = 0
}

let mydog = Dog()

mydog.name = "Rex"
print("name of dog: \(mydog.name)")
```

2.3.4.1 Add Database

จะเก็บลง Database ได้ด้วยคำสั่ง add คือการบันทึกลง database ซึ่งเก็บเป็นลักษณะ array แต่ทุกครั้งที่ต้องทำก่อน add คือ assign value ให้กับ object เสียก่อน หลักจากนั้นจึงสามารถบันทึกลง database ได้ ก่อนที่จะทำการบันทึก จำเป็นต้องประกาศตามโปรแกรม 3.2 เสียก่อน

โปรแกรม 2.4 Add Database

```
// Add Database
let realm = try! realm()

try! realm.write {
   realm.add(mydog)
}
```

2.3.4.2 Queries Filter

Query ด้วยคำสั่ง filter เป็นคำสั่งที่เป็นการกรองคัดเลือกจาก object ภายใน database โดยมีคำสั่งมากมายในการเรียกใช้ โดยโปรแกรม 3.3 จะเป็นตัวอย่างการใช้งาน

โปรแกรม 2.3 Realm Object

```
// Realm Object
class Dog: Object {
  dynamic var name = ""
  dynamic var age = 0
}

let mydog = Dog()

mydog.name = "Rex"
print("name of dog: \(mydog.name)")
```

2.3.4.1 Add Database

จะเก็บลง Database ได้ด้วยคำสั่ง add คือการบันทึกลง database ซึ่งเก็บเป็นลักษณะ array แต่ทุกครั้งที่ต้องทำก่อน add คือ assign value ให้กับ object เสียก่อน หลักจากนั้นจึงสามารถบันทึกลง database ได้ ก่อนที่จะทำการบันทึก จำเป็นต้องประกาศตามโปรแกรม 3.2 เสียก่อน

โปรแกรม 2.4 Add Database

```
// Add Database
let realm = try! realm()

try! realm.write {
  realm.add(mydog)
}
```

2.3.4.2 Queries Filter

Query ด้วยคำสั่ง filter เป็นคำสั่งที่เป็นการกรองคัดเลือกจาก object ภายใน database โดยมีคำสั่งมากมายในการเรียกใช้ โดยโปรแกรม 3.3 จะเป็นตัวอย่างการใช้งาน

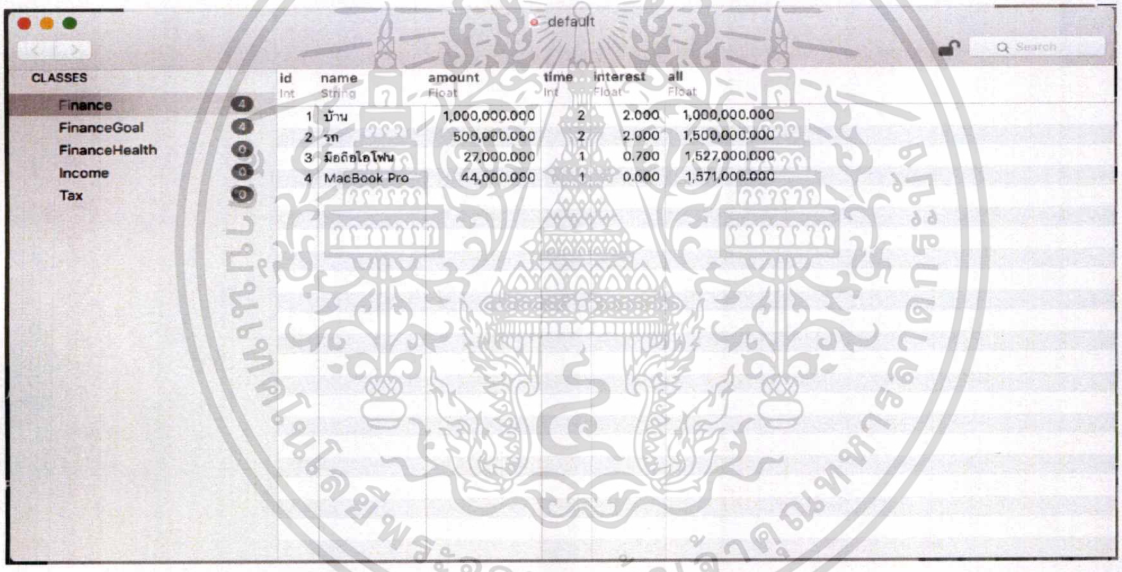
โปรแกรม 2.5 Queries Filter

```
// Queries Filter
let realm = try! Realm()
ley r = realm.object(Dog).filter("age > 8")

// Queries are chain able
let r2 = r.filter("name contains "rex")
```

2.4.3 Realm Browser Tool

เป็น Tools สำหรับผู้พัฒนา อนุญาตให้เข้าถึงง่ายและรวดเร็วในเนื้อหาของไฟล์ .realm สามารถที่จะปรับเปลี่ยนเนื้อหาของไฟล์ .realm โดยไม่จำเป็นต้องใช้รหัส ทำให้ง่ายต่อการ automatically generate เป็นรูปแบบ Realm Object Source Files ไว้สำหรับแก้ไขและดู



รูป 2.7 Realm Browser Tool

นี้คือตัวอย่างสำหรับการเรียกใช้ Realm Browser Tool โดยสามารถแก้ไขได้ผ่าน Tool ตัวนี้ วิธีเข้าใช้ ไฟล์ .realm ของแอพพลิเคชั่นมีวิธีดังนี้

1) เปิด Finder แล้วกด Command + Shift + G

2) หลังจากนั้นใส่ Path /Users/<username>/Library/Developer/CoreSimulator/Devices/<simulator-uuid>/data/Containers/Data/Application/<application-uuid>/Documents/default.realm

3) กด Enter จะพบไฟล์ .realm

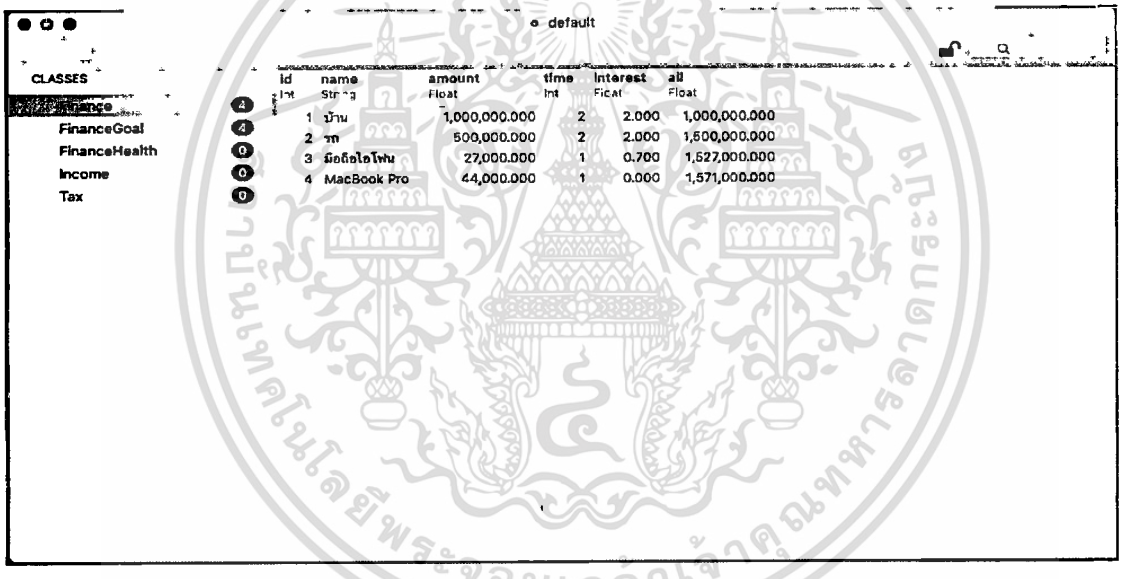
โปรแกรม 2.5 Queries Filter

```
// Queries Filter
let realm = try! Realm()
ley r = realm.object(Dog).filter("age > 8")

// Queries are chain able
let r2 = r.filter("name contains "rex")
```

2.4.3 Realm Browser Tool

เป็น Tools สำหรับผู้พัฒนา อนุญาตให้เข้าถึงง่ายและรวดเร็วในเนื้อหาของไฟล์ .realm สามารถที่จะปรับเปลี่ยนเนื้อหาของไฟล์ .realm โดยไม่จำเป็นต้องใช้รหัส ทำให้ง่ายต่อการ automatically generate เป็นรูปแบบ Realm Object Source Files ไว้สำหรับแก้ไขและดู



รูป 2.7 Realm Browser Tool

นี้คือตัวอย่างสำหรับการเรียกใช้ Realm Browser Tool โดยสามารถแก้ไขได้ผ่าน Tool ตัวนี้ วิธีเข้าใช้ ไฟล์ .realm ของแอพพลิเคชั่นมีวิธีดังนี้

1) เปิด Finder แล้วกด Command + Shift + G

2) หลังจากนั้นใส่ Path /Users/<username>/Library/Developer/CoreSimulator/Devices/<simulator-uuid>/data/Containers/Data/Application/<application-uuid>/Documents/default.realm

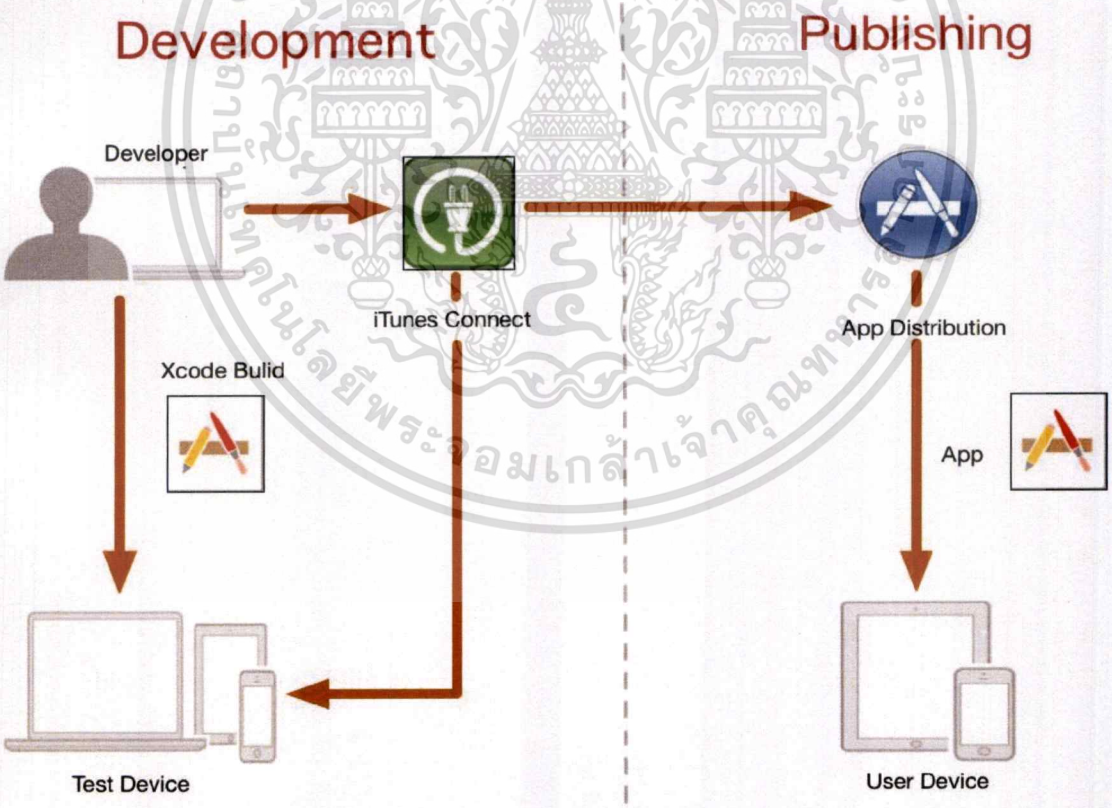
3) กด Enter จะพบไฟล์ .realm

บทที่ 3
การออกแบบและพัฒนา

3.1 ภาพรวมของการพัฒนา

ในการทำงานของตัวแอพพลิเคชั่น ผู้ใช้ต้องใช้งานผ่านระบบปฏิบัติการไอโอเอส (iOS) ผู้ใช้สามารถใส่รายละเอียดการเงินส่วนตัวบนแอพพลิเคชั่น โดยผู้ใช้สามารถเลือกประเภทของการเงินแต่ละส่วนได้ ซึ่งมีให้เลือกหลากหลายประเภท ผู้ใช้ยังสามารถเลือกดูกราฟการเงินส่วนบุคคลทั้งหมดได้ อีกทั้งยังมีคู่มือการใช้งานเบื้องต้น สำหรับผู้ใช้ที่ต้องการเข้าใจการทำงานเกี่ยวกับการเงินมากขึ้น

สำหรับส่วนแสดงผลที่ทำงานบนอุปกรณ์เคลื่อนที่ที่มีระบบปฏิบัติการไอโอเอส (iOS) โดยทำงานบนแอพพลิเคชั่น โดยแสดงผลผ่านหน้าจอบนแอพพลิเคชั่น สนับสนุนการเปลี่ยนอุปกรณ์หรือการใช้งานหลายอุปกรณ์ของผู้ใช้ โดยผ่านระบบไอคลาวด์ (iCloud)



รูป 3.1 ภาพรวมของการพัฒนา

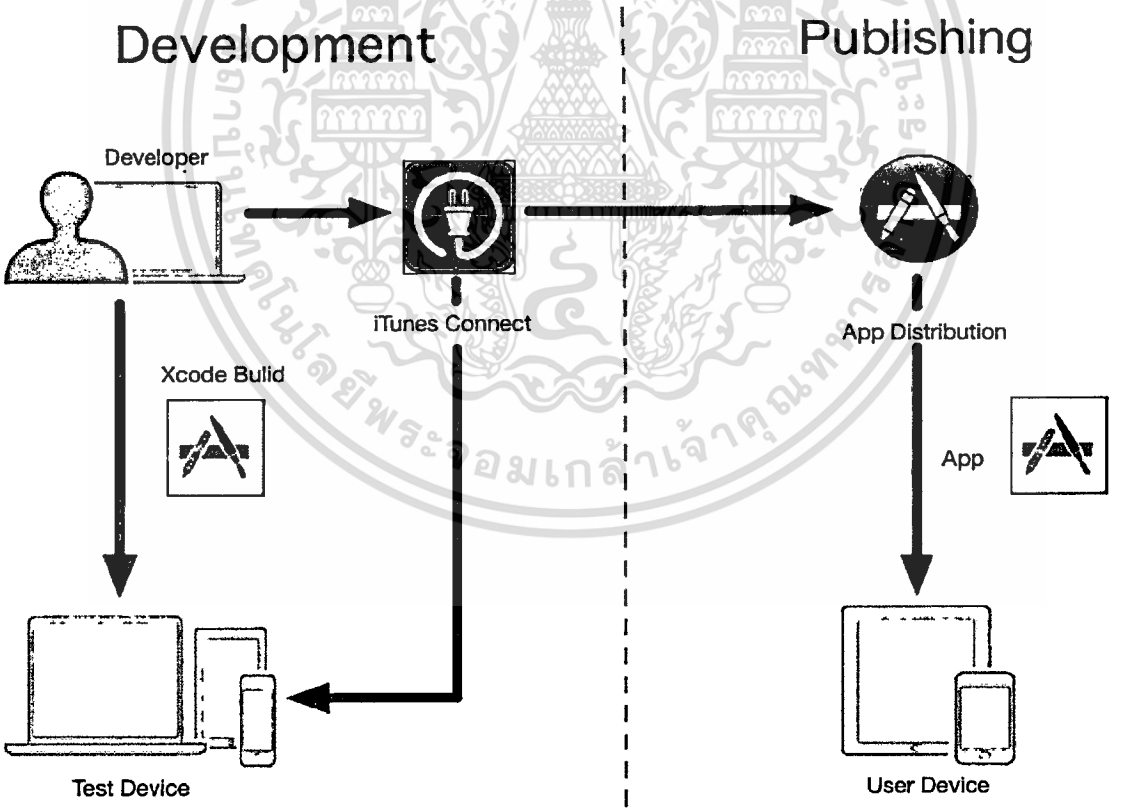
บทที่ 3

การออกแบบและพัฒนา

3.1 ภาพรวมของการพัฒนา

ในการทำงานของตัวแอพพลิเคชั่น ผู้ใช้ต้องใช้งานผ่านระบบปฏิบัติการไอโอเอส (iOS) ผู้ใช้สามารถใส่รายละเอียดการเงินส่วนตัวบนแอพพลิเคชั่น โดยผู้ใช้สามารถเลือกประเภทของการเงินแต่ละส่วนได้ ซึ่งมีให้เลือกหลากหลายประเภท ผู้ใช้ยังสามารถเลือกดูกราฟการเงินส่วนบุคคลทั้งหมดได้ อีกทั้งยังมีคู่มือการใช้งานเบื้องต้น สำหรับผู้ใช้ที่ต้องการเข้าใจการทำงานเกี่ยวกับการเงินมากขึ้น

สำหรับส่วนแสดงผลที่ทำงานบนอุปกรณ์เคลื่อนที่ที่มีระบบปฏิบัติการไอโอเอส (iOS) โดยทำงานบนแอพพลิเคชั่น โดยแสดงผลผ่านหน้าจอบนแอพพลิเคชั่น สนับสนุนการเปลี่ยนอุปกรณ์หรือการใช้งานหลายอุปกรณ์ของผู้ใช้ โดยผ่านระบบไอคลาวด์ (iCloud)



รูป 3.1 ภาพรวมของการพัฒนา

จากรูปที่ 3.1 แสดงถึงภาพรวมของระบบ โดยเริ่มจากผู้พัฒนาได้พัฒนาแอพพลิเคชั่นผ่านโปรแกรมเอ็กโค้ด (Xcode) หลังจากนั้นทำการต่อเข้ากับไอจูน (iTunes) เพื่อใช้ทดสอบบนตัวอุปกรณ์ที่มีระบบปฏิบัติการไอโอเอส (iOS)

การทำงานของระบบถูกแบ่งออกเป็น 2 ส่วนได้แก่ ส่วนแสดงผล (Front-end) และส่วนประมวลผล (Back-end) โดยแต่ละส่วนมีรายละเอียดดังนี้

3.2 เครื่องมือที่ใช้ในการพัฒนา

การพัฒนาแอพพลิเคชั่นผ่านระบบปฏิบัติการไอโอเอส (iOS) มีการกำหนดเครื่องมือและภาษาที่ใช้ในการพัฒนาเพื่อให้มีความเหมาะสมกับระบบ และความต้องการใช้งานดังนี้

3.2.1 ระบบปฏิบัติการ

1) สามารถทำงานได้บนระบบปฏิบัติการไอโอเอส (iOS) ที่มีเวอร์ชั่นตั้งแต่ 8.0 ขึ้นไป

2) พัฒนาแอพพลิเคชั่น ที่มีระบบปฏิบัติการโอเอสเอ็ก (OS X) โดยเลือกใช้โอเอสเอ็ก (OS X) เวอร์ชั่น El Capitan 10.11.1

3.2.2 ภาษาโปรแกรม

ภาษาที่ใช้ในการพัฒนาแอพพลิเคชั่นผ่านระบบปฏิบัติไอโอเอส สามารถแบ่งได้ตามลักษณะการทำงาน 2 ส่วน คือ ส่วนการพัฒนาแอพพลิเคชั่นบนสมาร์ทโฟน และส่วนการส่งข้อมูลระหว่างแอพพลิเคชั่นบนสมาร์ทโฟนกับฐานข้อมูล ซึ่งสามารถสรุปภาษาที่ใช้ในการพัฒนาได้ดังนี้

1) ภาษาสวิฟต์ (Swift) สำหรับพัฒนาการทำงานทั้งหมดของระบบปฏิบัติการไอโอเอส

2) เรียลดาต้าเบส (Realm Database) สำหรับจัดการกับฐานข้อมูลของระบบ

3.2.3 เครื่องมือที่ใช้ในการพัฒนา

1) เอ็กโค้ด (Xcode) เป็นเครื่องมือในการพัฒนาไอโอเอส (iOS) ในส่วนขอแอพพลิเคชั่น

2) ออมนิกราฟเฟิล (OmniGraffle) เป็นเครื่องมือในการออกแบบส่วนของหน้าต่างผู้ใช้งาน

3) สเก็ต (Sketch) เป็นเครื่องมือในการออกแบบส่วนของแอพพลิเคชั่น ไอคอน

3.3 รูปแบบโครงสร้างของโปรแกรม (System Requirment)

3.3.1 Functional Requirement

1) ระบบระบุเป้าหมายการเงินเพื่อออมเงินในแต่ละเดือน

2) ระบบแสดงข้อมูลในรูปแบบกราฟ

3) ระบบคำนวณภาษี

4) ระบบคำนวณกองทุน

จากรูปที่ 3.1 แสดงถึงภาพรวมของระบบ โดยเริ่มจากผู้พัฒนาได้พัฒนาแอพพลิเคชั่นผ่านโปรแกรมเอ็กโค้ด (Xcode) หลังจากนั้นทำการต่อเข้ากับไอจูน (iTunes) เพื่อใช้ทดสอบบนตัวอุปกรณ์ที่มีระบบปฏิบัติการไอโอเอส (iOS)

การทำงานของระบบถูกแบ่งออกเป็น 2 ส่วนได้แก่ ส่วนแสดงผล (Front-end) และส่วนประมวลผล (Back-end) โดยแต่ละส่วนมีรายละเอียดดังนี้

3.2 เครื่องมือที่ใช้ในการพัฒนา

การพัฒนาแอพพลิเคชั่นผ่านระบบปฏิบัติการไอโอเอส (iOS) มีการกำหนดเครื่องมือและภาษาที่ใช้ในการพัฒนาเพื่อให้มีความเหมาะสมกับระบบ และความต้องการใช้งานดังนี้

3.2.1 ระบบปฏิบัติการ

1) สามารถทำงานได้บนระบบปฏิบัติการไอโอเอส (iOS) ที่มีเวอร์ชั่นตั้งแต่ 8.0 ขึ้นไป

2) พัฒนาแอพพลิเคชั่น ที่มีระบบปฏิบัติการโอเอสเอ็ก (OS X) โดยเลือกใช้โอเอสเอ็ก (OS X) เวอร์ชั่น El Capitan 10.11.1

3.2.2 ภาษาโปรแกรม

ภาษาที่ใช้ในการพัฒนาแอพพลิเคชั่นผ่านระบบปฏิบัติไอโอเอส สามารถแบ่งได้ตามลักษณะการทำงาน 2 ส่วน คือ ส่วนการพัฒนาแอพพลิเคชั่นบนสมาร์ทโฟน และส่วนการส่งข้อมูลระหว่างแอพพลิเคชั่นบนสมาร์ทโฟนกับฐานข้อมูล ซึ่งสามารถสรุปภาษาที่ใช้ในการพัฒนาได้ดังนี้

1) ภาษาสวิฟต์ (Swift) สำหรับพัฒนาการทำงานทั้งหมดของระบบปฏิบัติการไอโอเอส

2) เรียลดาต้าเบส (Realm Database) สำหรับจัดการกับฐานข้อมูลของระบบ

3.2.3 เครื่องมือที่ใช้ในการพัฒนา

1) เอ็กโค้ด (Xcode) เป็นเครื่องมือในการพัฒนาไอโอเอส (iOS) ในส่วนขอแอพพลิเคชั่น

2) ออมนิกราฟเฟิล (OmniGraffle) เป็นเครื่องมือในการออกแบบส่วนของหน้าต่างผู้ใช้งาน

3) สเก็ต (Sketch) เป็นเครื่องมือในการออกแบบส่วนของแอพพลิเคชั่น ไอคอน

3.3 รูปแบบโครงสร้างของโปรแกรม (System Requirment)

3.3.1 Functional Requirement

1) ระบบระบุเป้าหมายการเงินเพื่อออมเงินในแต่ละเดือน

2) ระบบแสดงข้อมูลในรูปแบบกราฟ

3) ระบบคำนวณภาษี

4) ระบบคำนวณกองทุน

5) คำอธิบายเพิ่มเติมให้อ่าน เพื่อให้เข้าใจการใช้งานง่าย

3.3.2 Non - Functional Requirement

1) แอพพลิเคชั่นมีหน้าตาทันสมัย

2) แอพพลิเคชั่นใช้งานง่าย ไม่ซับซ้อน

3) แอพพลิเคชั่นมีการรักษาความปลอดภัยข้อมูลต่างๆ

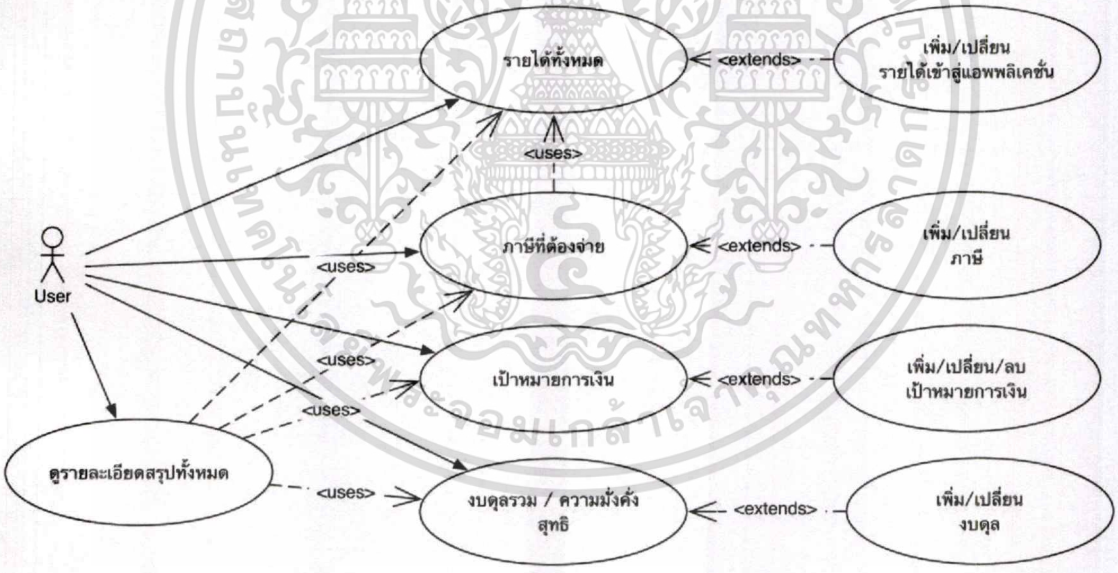
4) แอพพลิเคชั่นมีการสำรองข้อมูลที่ผู้ใช้งานบันทึกไว้ทั้งหมดแล้ว

5) แอพพลิเคชั่นมีการอธิบายการใช้งาน

3.4 แผนภาพ UML (Unified Modelling Language Diagram)

3.4.1 Use Case Diagram

จุดประสงค์หลักของการเขียนยูสเคสไดอะแกรม (Use Case Diagram) ก็เพื่อเล่าเรื่องราวของปัญหาทั้งหมดว่า มีส่วนประกอบอะไรบ้างและเกี่ยวพันกันจนกลายเป็นระบบได้อย่างไร การเขียนยูสเคส ไดอะแกรมจะช่วยให้ผู้พัฒนาระบบสามารถแยกแยะได้ว่าจะมีกิจกรรมอะไรที่น่าจะเกิดขึ้นในระบบบ้าง ดังรูป 3.2



รูป 3.2 Use Case Diagram

จากแผนภาพยูสเคส สามารถอธิบายได้ดังต่อไปนี้

1) ภาษี ผู้ใช้สามารถคำนวณภาษี โดยใส่รายละเอียดรายได้ - รายจ่ายของ ผู้ใช้ เมื่อใส่รายละเอียดเสร็จสิ้น ระบบจะคำนวณ จำนวนลดหย่อน LTF, RMF

5) คำอธิบายเพิ่มเติมให้อ่าน เพื่อให้เข้าใจการใช้งานง่าย

3.3.2 Non - Functional Requirement

1) แอพพลิเคชั่นมีหน้าตาทันสมัย

2) แอพพลิเคชั่นใช้งานง่าย ไม่ซับซ้อน

3) แอพพลิเคชั่นมีการรักษาความปลอดภัยข้อมูลต่างๆ

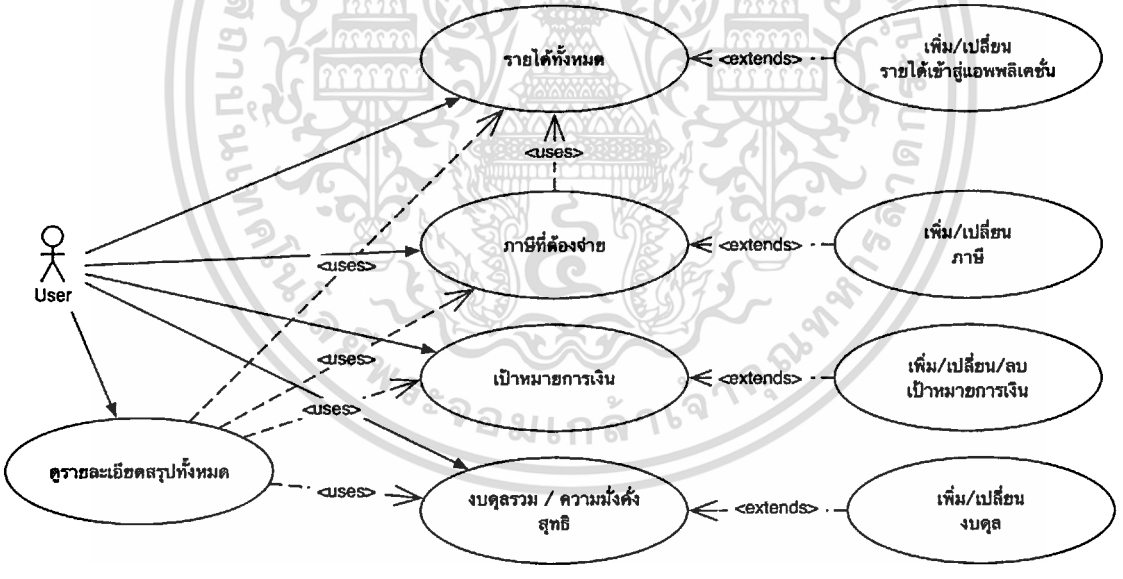
4) แอพพลิเคชั่นมีการสำรองข้อมูลที่ผู้ใช้งานบันทึกไว้ทั้งหมดแล้ว

5) แอพพลิเคชั่นมีการอธิบายการใช้งาน

3.4 แผนภาพ UML (Unified Modelling Language Diagram)

3.4.1 Use Case Diagram

จุดประสงค์หลักของการเขียนยูสเคสไดอะแกรม (Use Case Diagram) ก็เพื่อเล่าเรื่องราวของปัญหาทั้งหมดว่า มีส่วนประกอบอะไรบ้างและเกี่ยวพันกันจนกลายเป็นระบบได้อย่างไร การเขียนยูสเคส ไดอะแกรมจะช่วยให้ผู้พัฒนาระบบสามารถแยกแยะได้ว่าจะมีกิจกรรมอะไรที่น่าจะเกิดขึ้นในระบบบ้าง ดังรูป 3.2



รูป 3.2 Use Case Diagram

จากแผนภาพยูสเคส สามารถอธิบายได้ดังต่อไปนี้

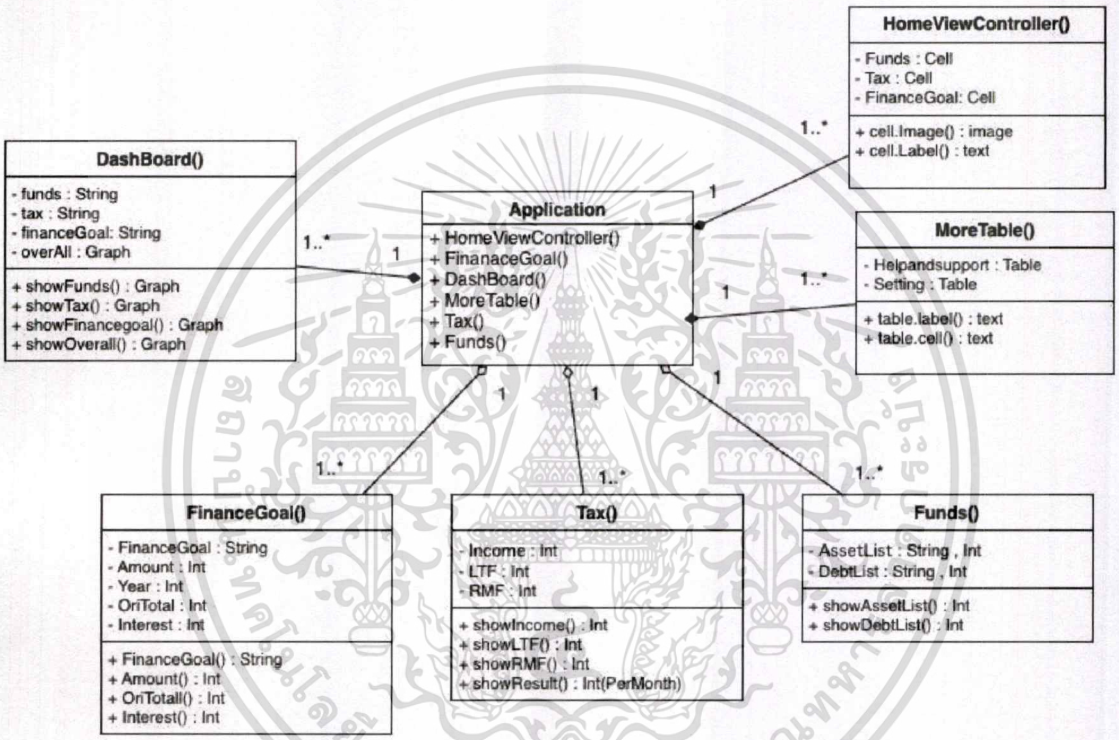
1) ภาษี ผู้ใช้สามารถคำนวณภาษี โดยใส่รายละเอียดรายได้ - รายจ่ายของ
ผู้ใช้ เมื่อใส่รายละเอียดเสร็จสิ้น ระบบจะคำนวณ จำนวนลดหย่อน LTF, RMF

2) เป้าหมายการเงิน ผู้ใช้สามารถกำหนดเป้าหมายการเงินได้โดยเลือกเป้าหมายที่ต้องการเช่น รถ , บ้าน เป็นต้น และจำนวนเงิน ระยะเวลา ดอกเบี้ย ระบบจะคำนวณ จำนวนเงินที่ต้องเก็บต่อเดือน

3) งบดุล หรือสุขภาพการเงิน ผู้ใช้สามารถเลือกสินทรัพย์แต่ละชนิด แล้วระบบจะวิเคราะห์ในแต่ละส่วนให้ผู้ใช้ได้เห็น

4) สรุป หรือส่วนแสดงข้อมูลทั้งหมดออกเป็นรูปแบบกราฟ

3.4.2 Class Diagram



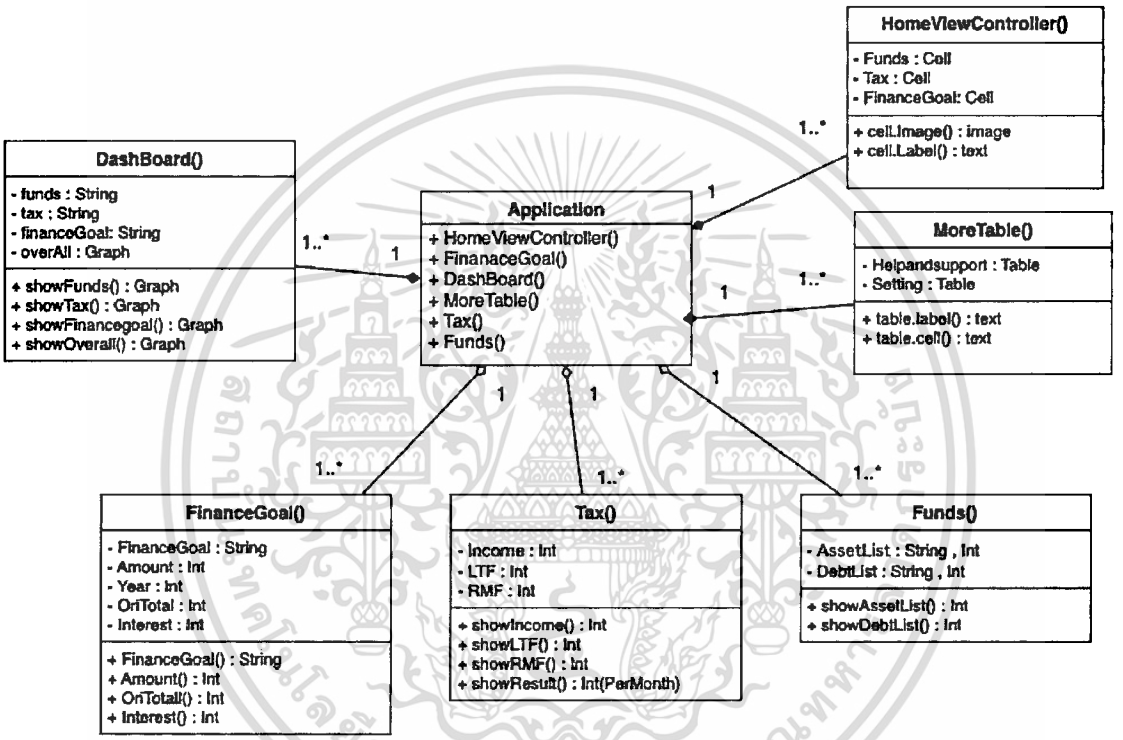
รูป 3.3 Class Diagram

2) เป้าหมายการเงิน ผู้ใช้สามารถกำหนดเป้าหมายการเงินได้โดยเลือกเป้าหมายที่ต้องการเช่น รถ , บ้าน เป็นต้น และจำนวนเงิน ระยะเวลา ดอกเบี้ย ระบบจะคำนวณ จำนวนเงินที่ต้องเก็บต่อเดือน

3) งบดุล หรือสุขภาพการเงิน ผู้ใช้สามารถเลือกสินทรัพย์แต่ละชนิด แล้วระบบจะวิเคราะห์ในแต่ละส่วนให้ผู้ใช้ได้เห็น

4) สรุป หรือส่วนแสดงข้อมูลทั้งหมดออกเป็นรูปแบบกราฟ

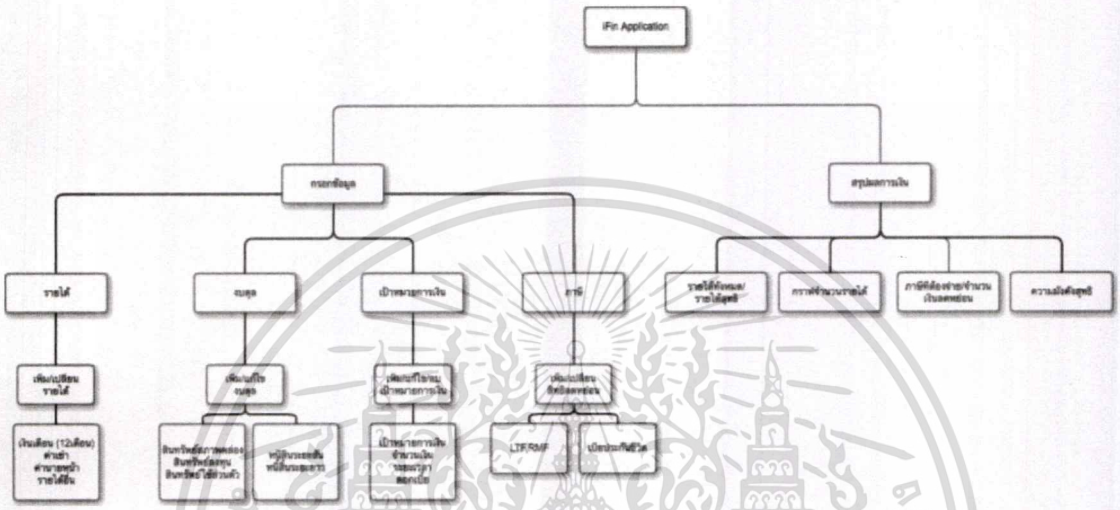
3.4.2 Class Diagram



รูป 3.3 Class Diagram

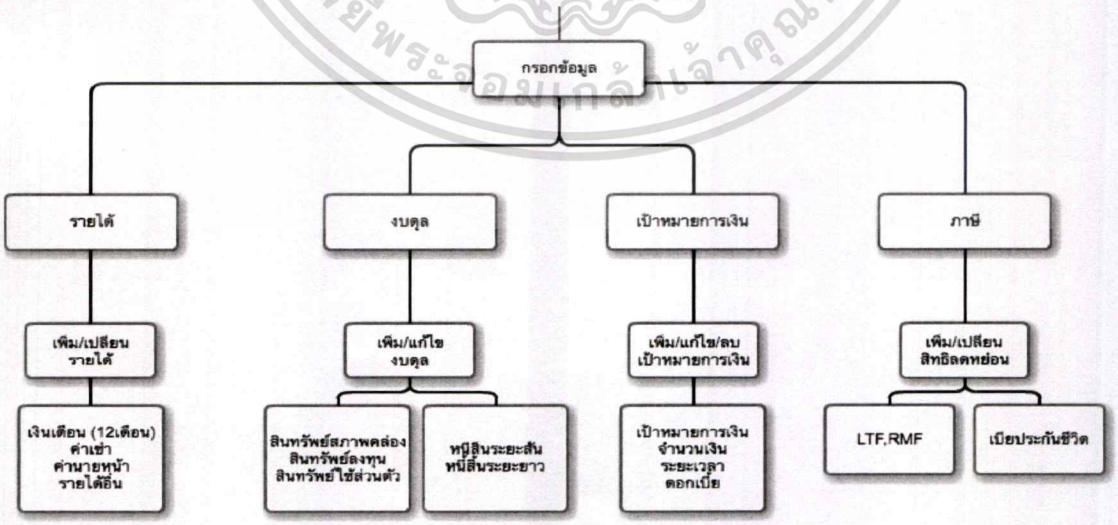
3.4.3 Top-Down Diagram

อธิบายความสัมพันธ์ภาพรวมของระบบ โดยที่แอพพลิเคชั่นจะมีอยู่สองส่วนหลัก โดยส่วนแรกจะแสดงความสัมพันธ์ของการกรอกข้อมูลลงบันตัวแอพพลิเคชั่น โดยความสัมพันธ์เหล่านี้จะแสดงถึง ว่าแต่ละส่วนประกอบไปด้วยอะไรบ้าง ส่วนที่สองจะแสดงความสัมพันธ์ระหว่างหน้าต่างที่แสดงผลการทำงานนทั้งหมด โดยองค์ประกอบทั้งหมดแสดงดังรูป 3.4



รูป 3.4 Top - Down Diagram

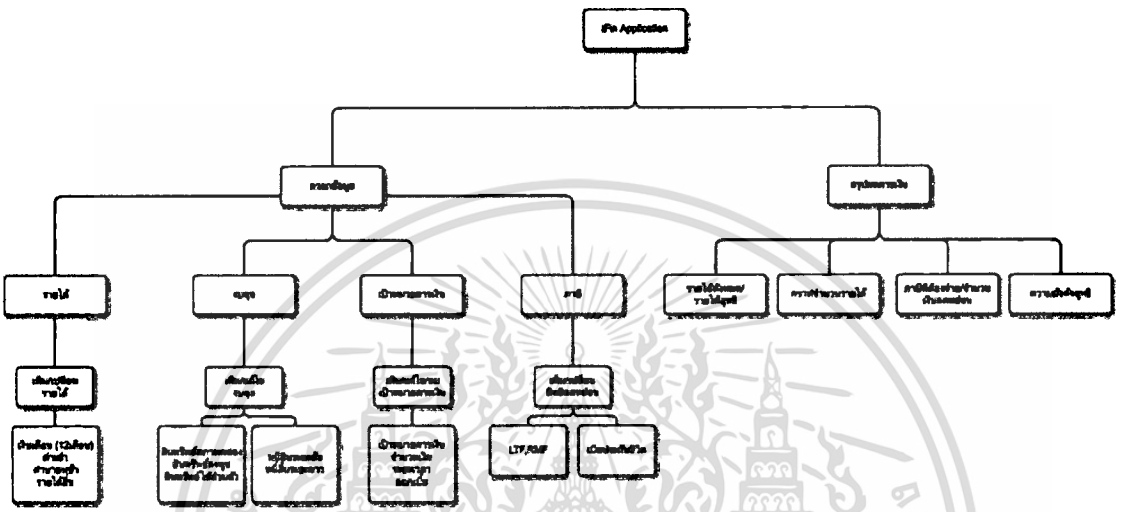
ในส่วนแรกแบ่งย่อยได้ดังรูป 3.5



รูป 3.5 Input Top - Down Diagram

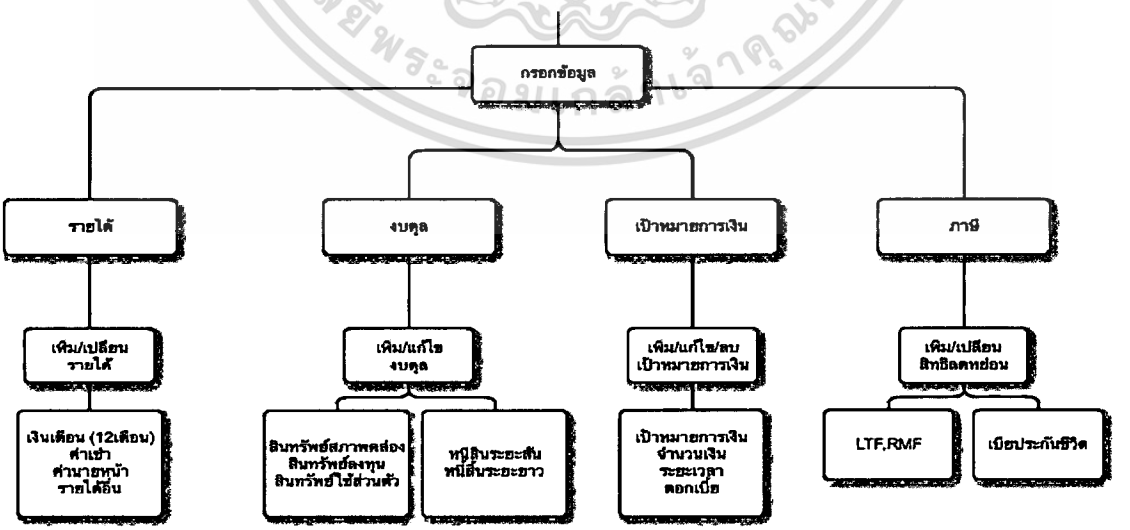
3.4.3 Top-Down Diagram

อธิบายความสัมพันธ์ภาพรวมของระบบ โดยที่แอพพลิเคชั่นจะมีอยู่สองส่วนหลัก โดยส่วนแรกจะแสดงความสัมพันธ์ของการกรอกข้อมูลลงบันตัวแอพพลิเคชั่น โดยความสัมพันธ์เหล่านี้จะแสดงถึง ว่าแต่ละส่วนประกอบไปด้วยอะไรบ้าง ส่วนที่สองจะแสดงความสัมพันธ์ระหว่างหน้าต่างที่แสดงผลการทำงานนทั้งหมด โดยองค์ประกอบทั้งหมดแสดงดังรูป 3.4



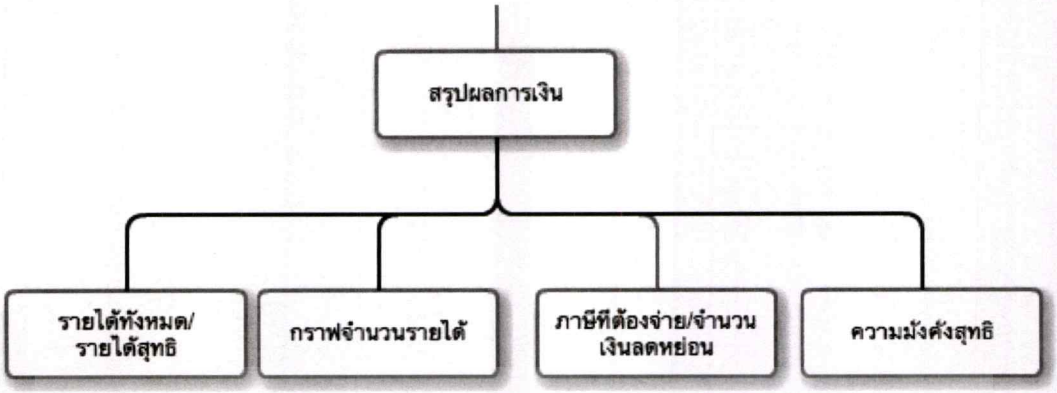
รูป 3.4 Top - Down Diagram

ในส่วนแรกแบ่งย่อยได้ดังรูป 3.5



รูป 3.5 Input Top - Down Diagram

ในส่วนที่สองแบ่งย่อยได้ดังรูป 3.6



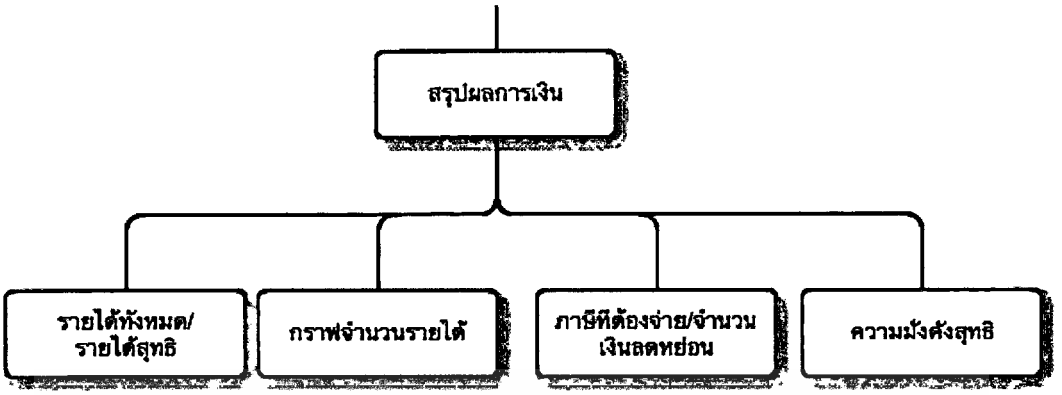
รูป 3.6 Dashboard Top - Down Diagram

3.4.4 Activity Diagram

Activity Diagram หรือแผนภาพกิจกรรม ใช้อธิบายกิจกรรมที่เกิดขึ้นในลักษณะกระแสการไหลของการทำงาน (Workflow) จะมีลักษณะเดียวกับ Flowchart (แสดงขั้นตอนการทำงานของระบบ) โดยขั้นตอนในการทำงานแต่ละขั้นจะเรียกว่า Activity การทำงานหลักดังนี้ การคำนวณผลลัพธ์บางอย่าง, การเปลี่ยนแปลงสถานะ (State) ของระบบ, การส่งค่ากลับคืน, การส่งสัญญาณ, การเรียกใช้ Operation (Method) อื่นๆ เพื่อทำงาน, การสร้าง หรือ ทำลายวัตถุ

ส่วนของ Activity Diagram ที่ออกแบบจะมีอยู่ 5 ส่วนด้วยกัน

ในส่วนที่สองแบ่งย่อยได้ดังรูป 3.6



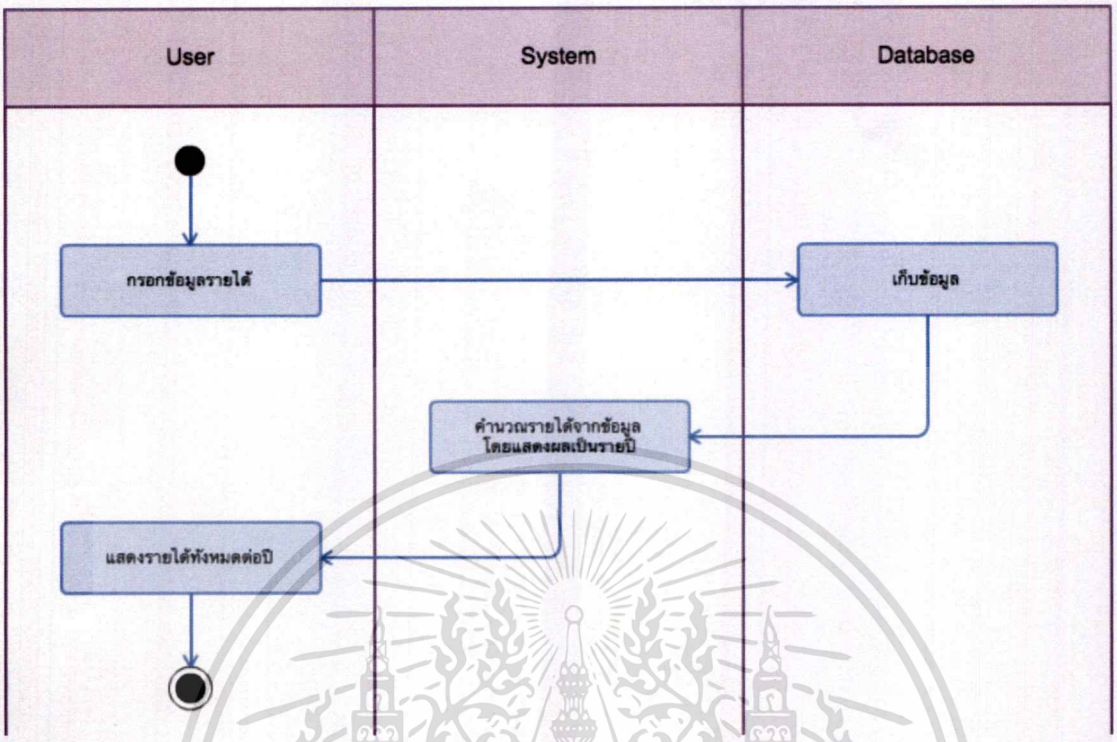
รูป 3.6 Dashboard Top - Down Diagram

3.4.4 Activity Diagram

Activity Diagram หรือแผนภาพกิจกรรม ใช้อธิบายกิจกรรมที่เกิดขึ้นในลักษณะกระแสการไหลของการทำงาน (Workflow) จะมีลักษณะเดียวกับ Flowchart (แสดงขั้นตอนการทำงานของระบบ) โดยขั้นตอนในการทำงานแต่ละขั้นจะเรียกว่า Activity การทำงานหลักดังนี้ การคำนวณผลลัพธ์บางอย่าง, การเปลี่ยนแปลงสถานะ (State) ของระบบ, การส่งค่ากลับคืน, การส่งสัญญาณ, การเรียกใช้ Operation (Method) อื่นๆ เพื่อทำงาน, การสร้าง หรือ ทำลายวัตถุ

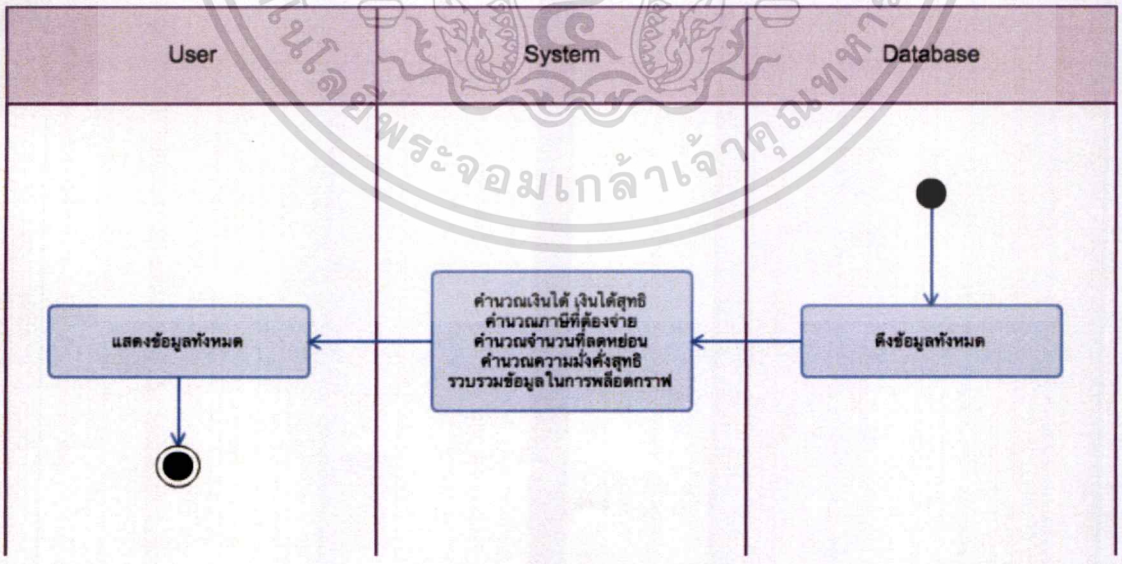
ส่วนของ Activity Diagram ที่ออกแบบจะมีอยู่ 5 ส่วนด้วยกัน

ส่วนที่ 1 คือ Activity Diagram “รายได้”



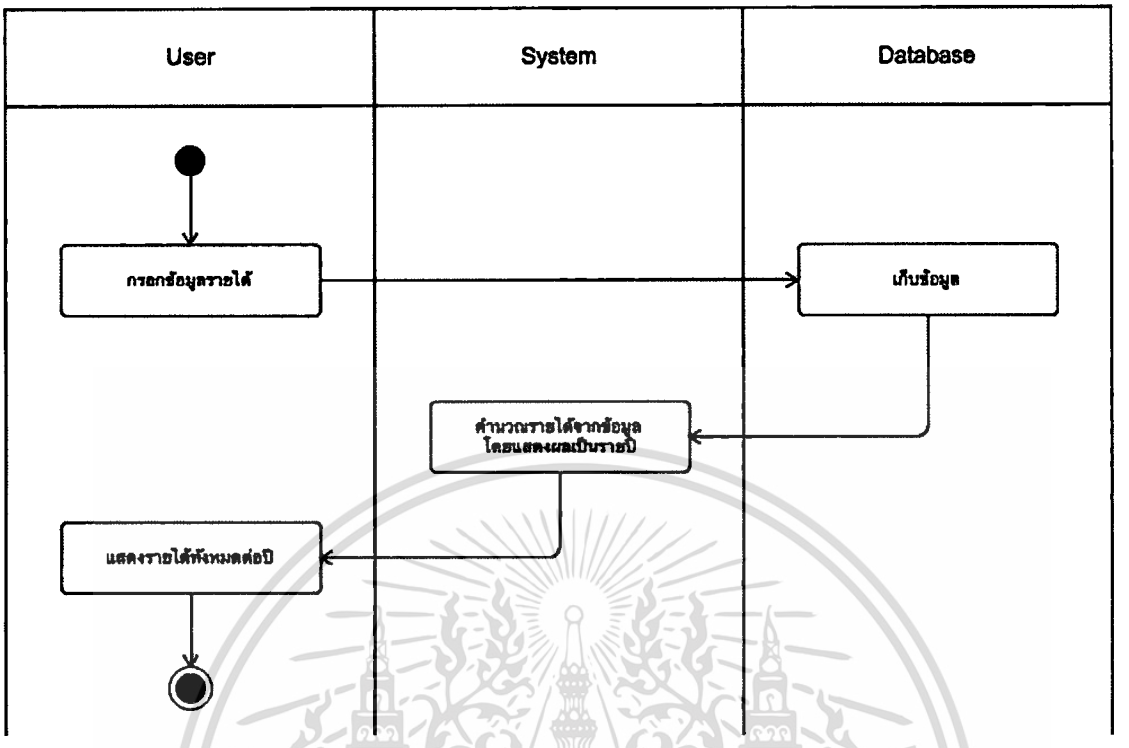
รูป 3.7 Activity Diagram Income

ส่วนที่ 2 คือ Activity Digram “สรุป”



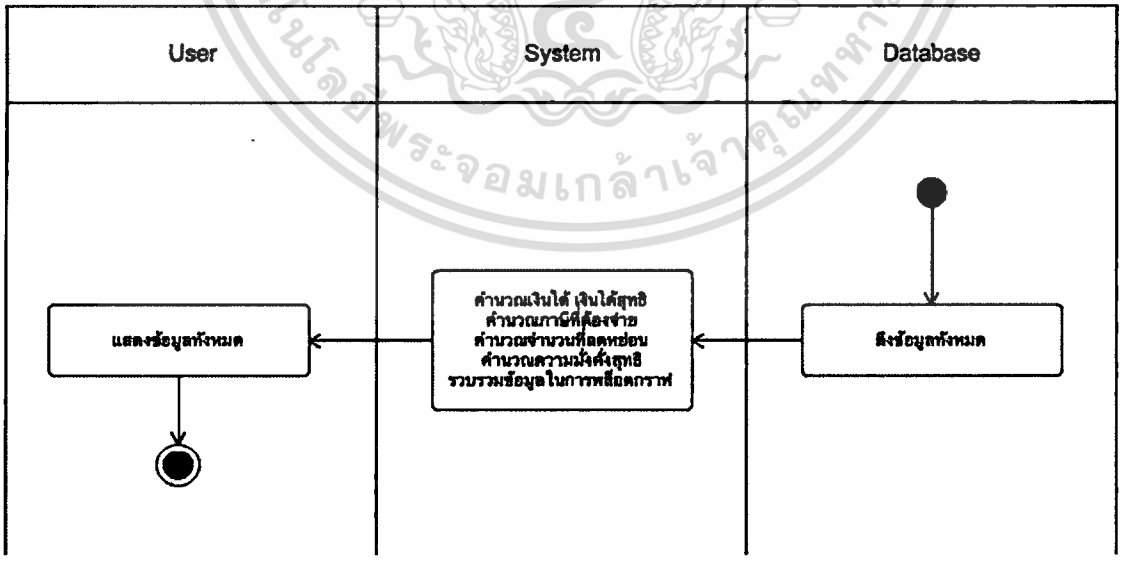
รูป 3.8 Activity Diagram Dashboard

ส่วนที่ 1 คือ Activity Diagram “รายได้”



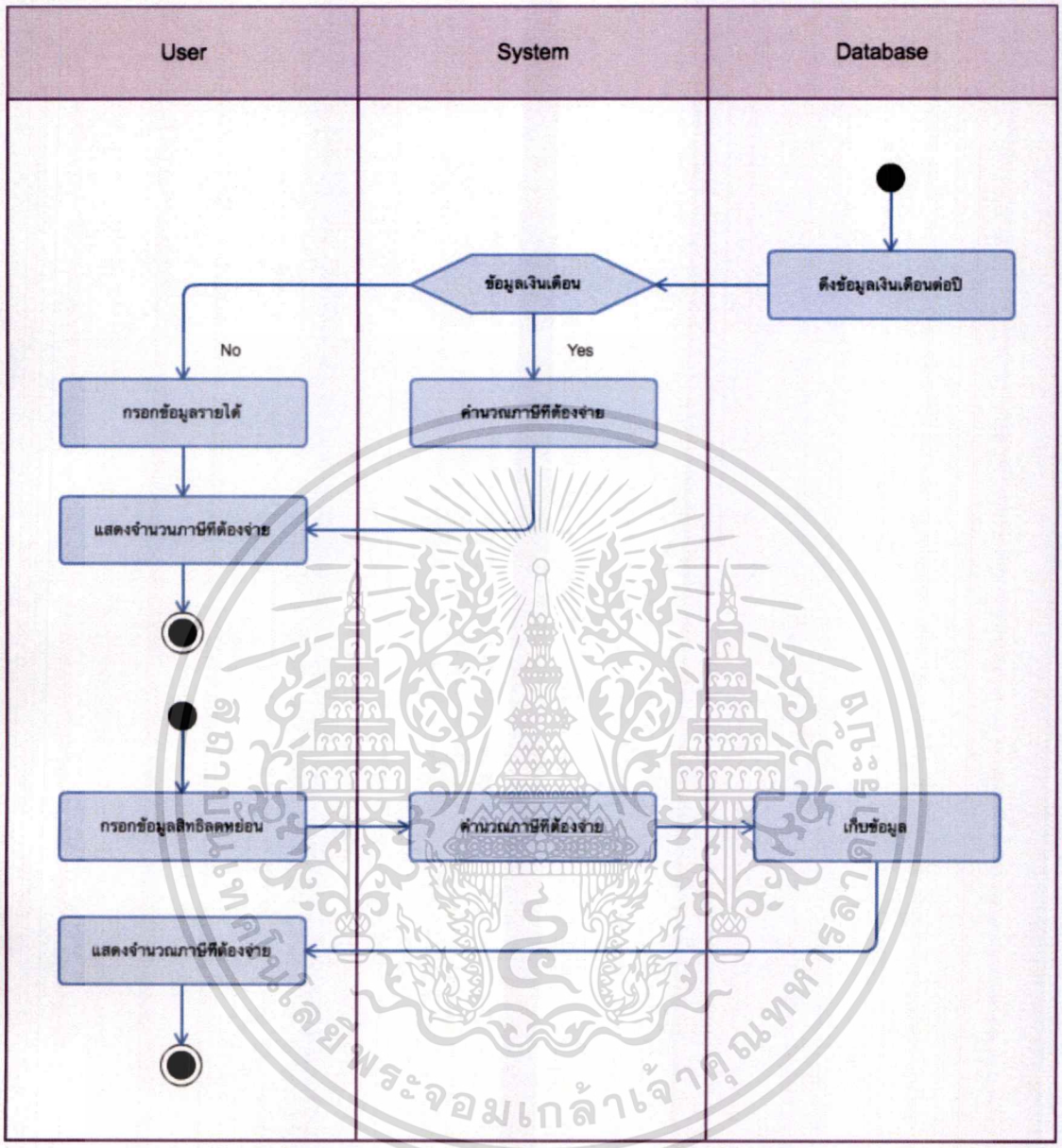
รูป 3.7 Activity Diagram Income

ส่วนที่ 2 คือ Activity Digram “สรุป”



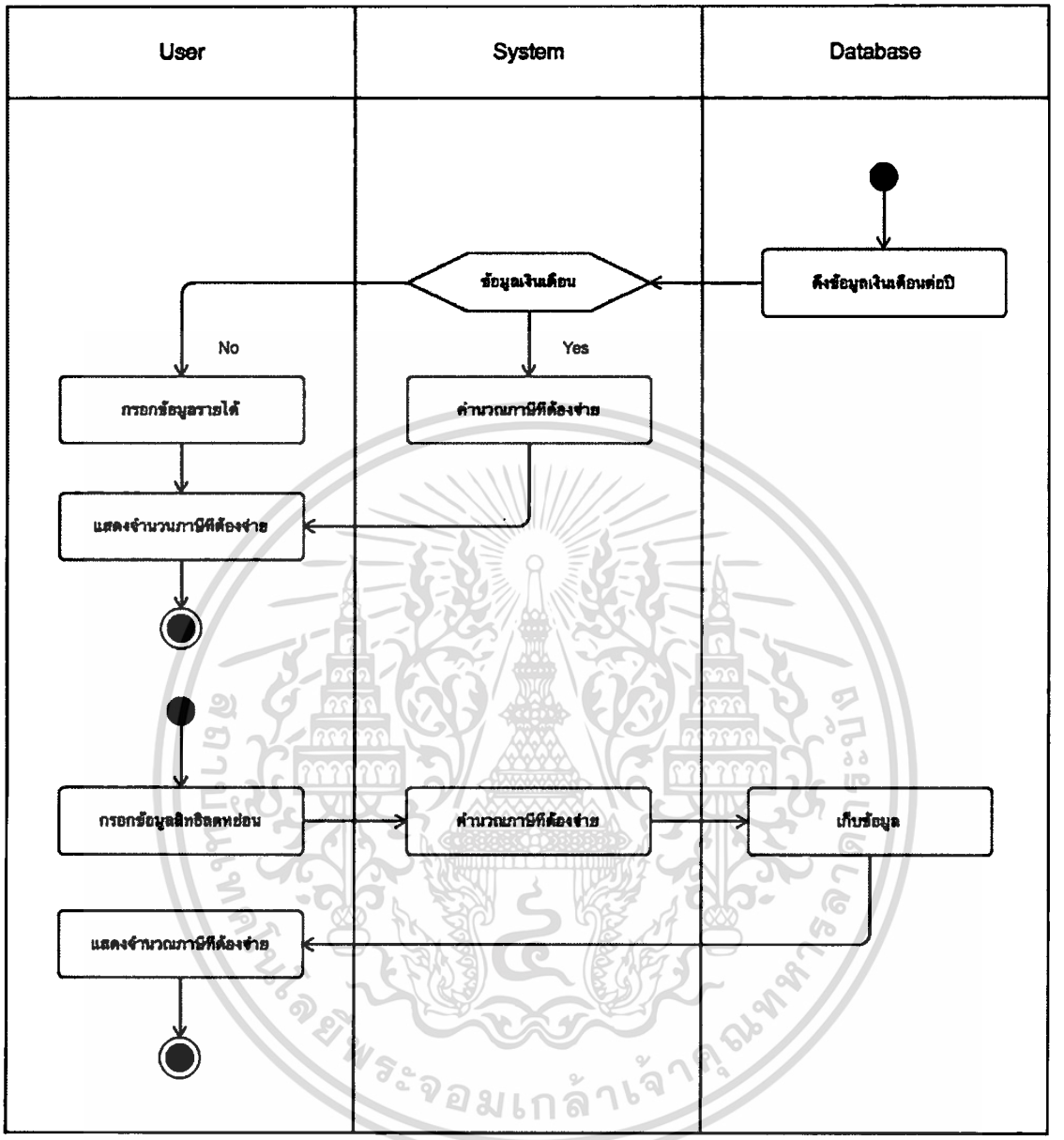
รูป 3.8 Activity Diagram Dashboard

ส่วนที่ 3 คือ Activity Diagram “ภาษี”



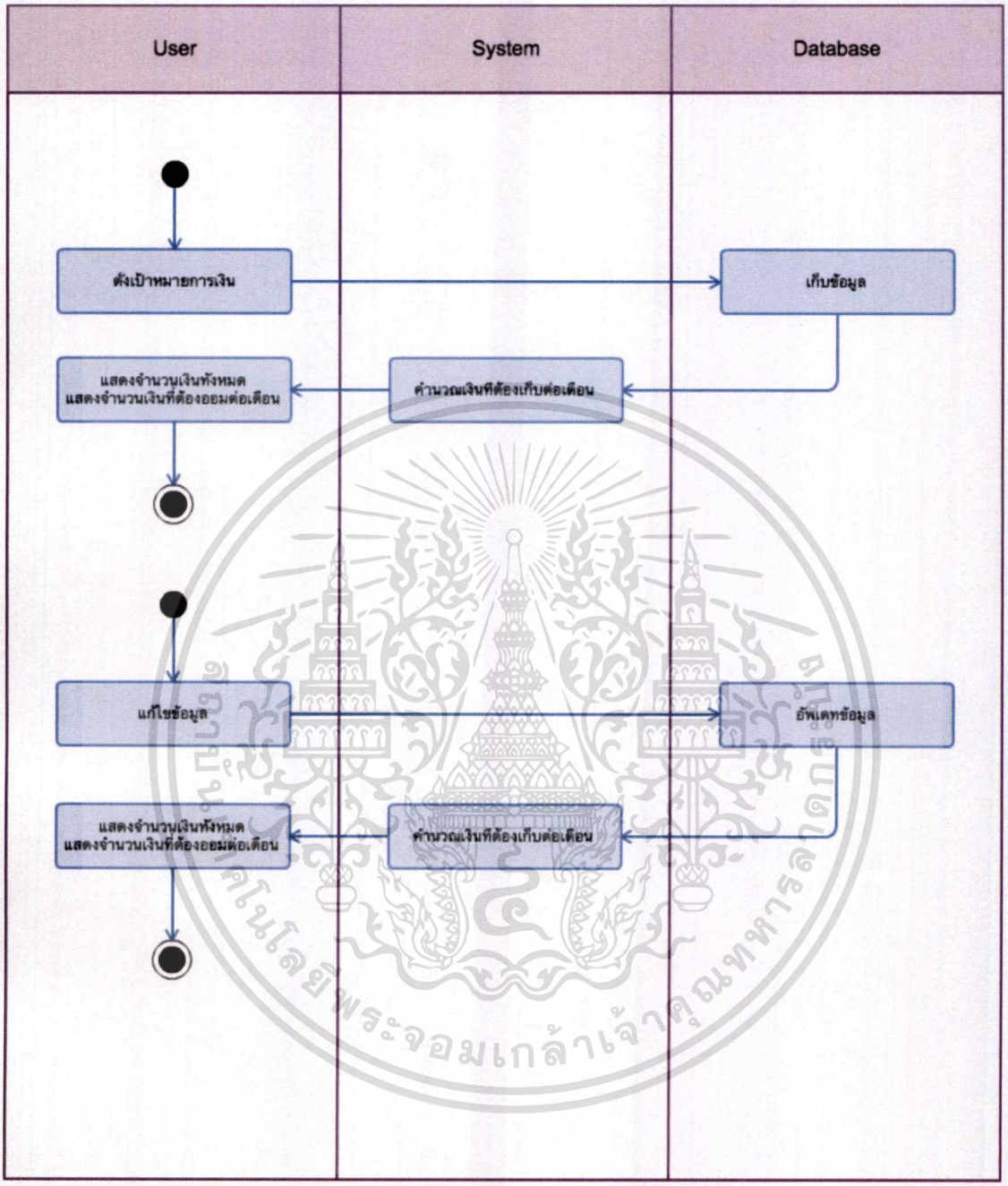
รูป 3.9 Activity Diagram Tax

ส่วนที่ 3 คือ Activity Diagram “ภาษี”



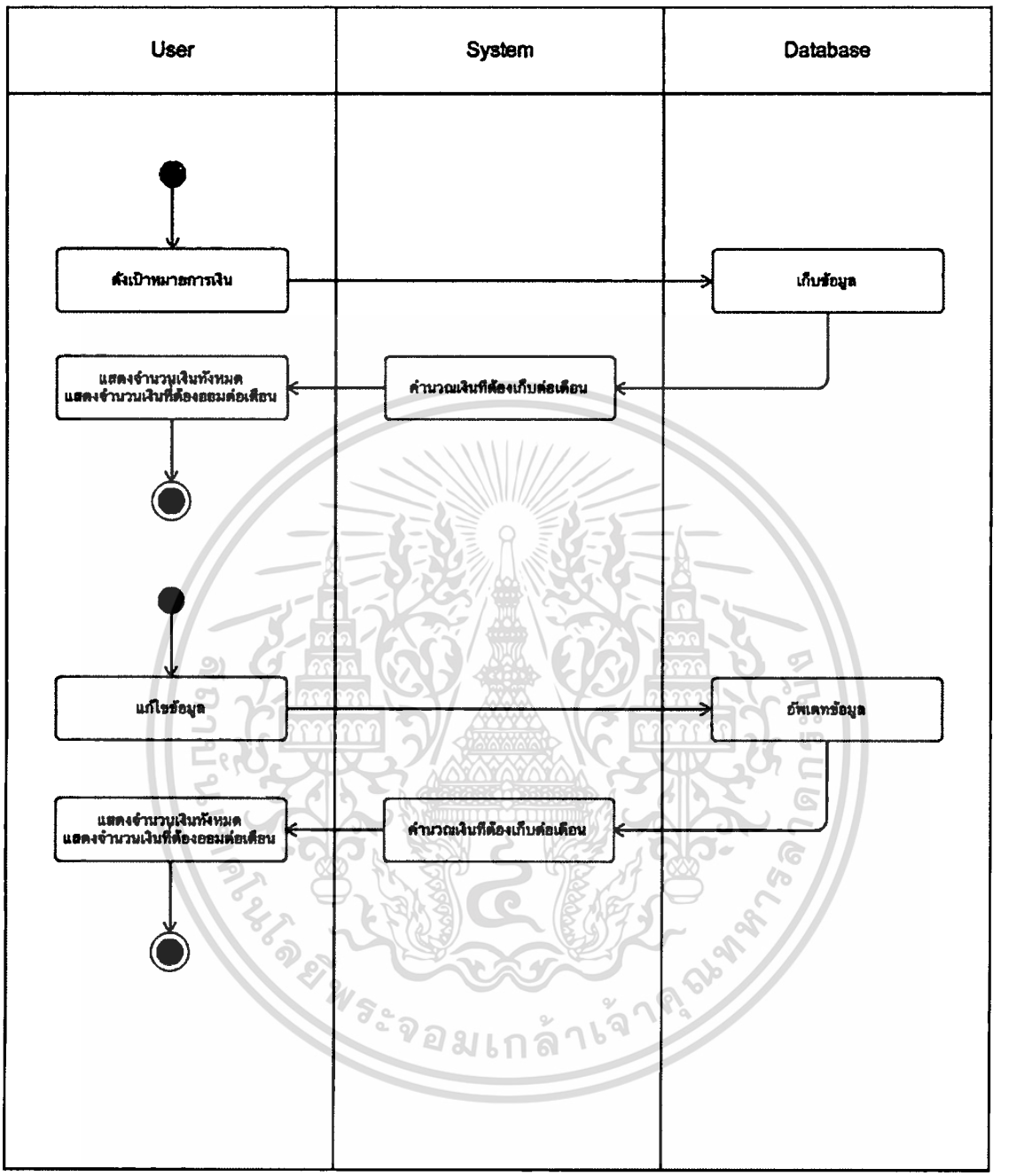
รูป 3.9 Activity Diagram Tax

ส่วนที่ 4 คือ Activity Diagram “เป้าหมายการเงิน”



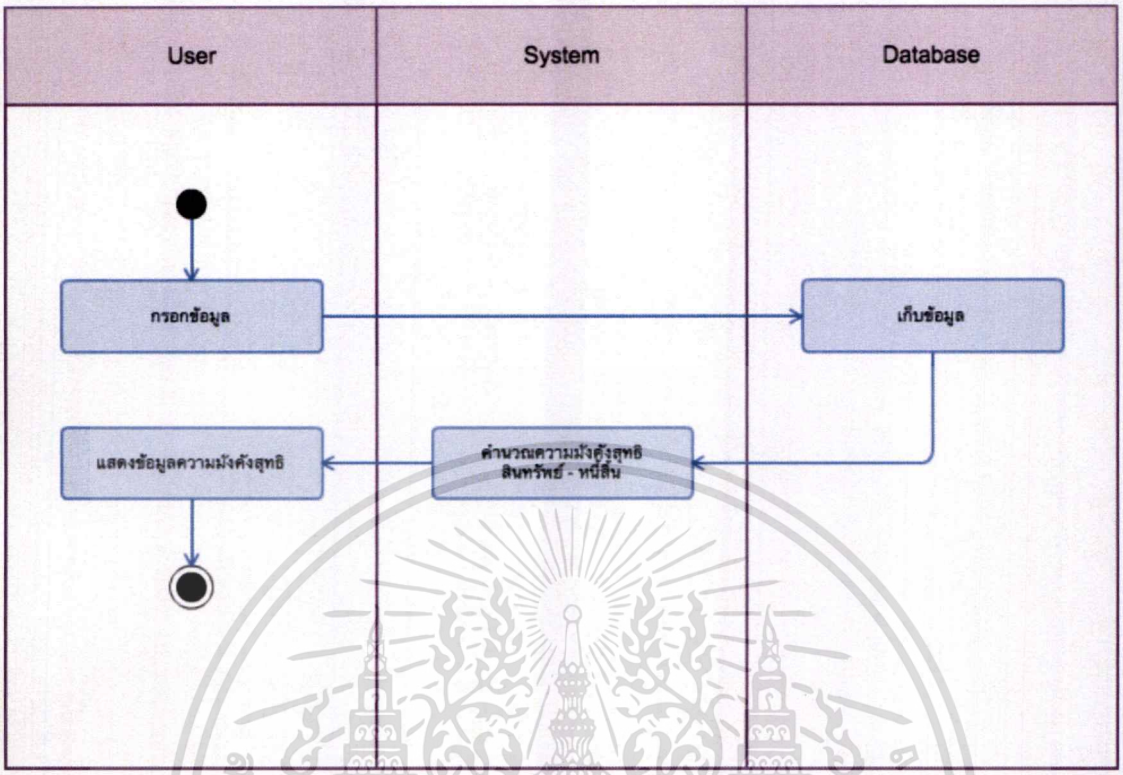
รูป 3.10 Activity Diagram Finance Goal

ส่วนที่ 4 คือ Activity Diagram “เป้าหมายการเงิน”



รูป 3.10 Activity Diagram Finance Goal

ส่วนที่ 5 คือ Activity Diagram “ตรวจสุขภาพการเงิน”



รูป 3.11 Activity Diagram Finance Health

3.5 ส่วนการเก็บฐานข้อมูล

ส่วนการเก็บฐานข้อมูลจะกล่าวถึงองค์ประกอบของส่วนต่อประสานกับผู้ใช้ และการส่งข้อมูลระหว่างส่วนต่อประสานกับผู้ใช้บนไอโอเอส (iOS) และฐานข้อมูลเรียล (Realm Database)

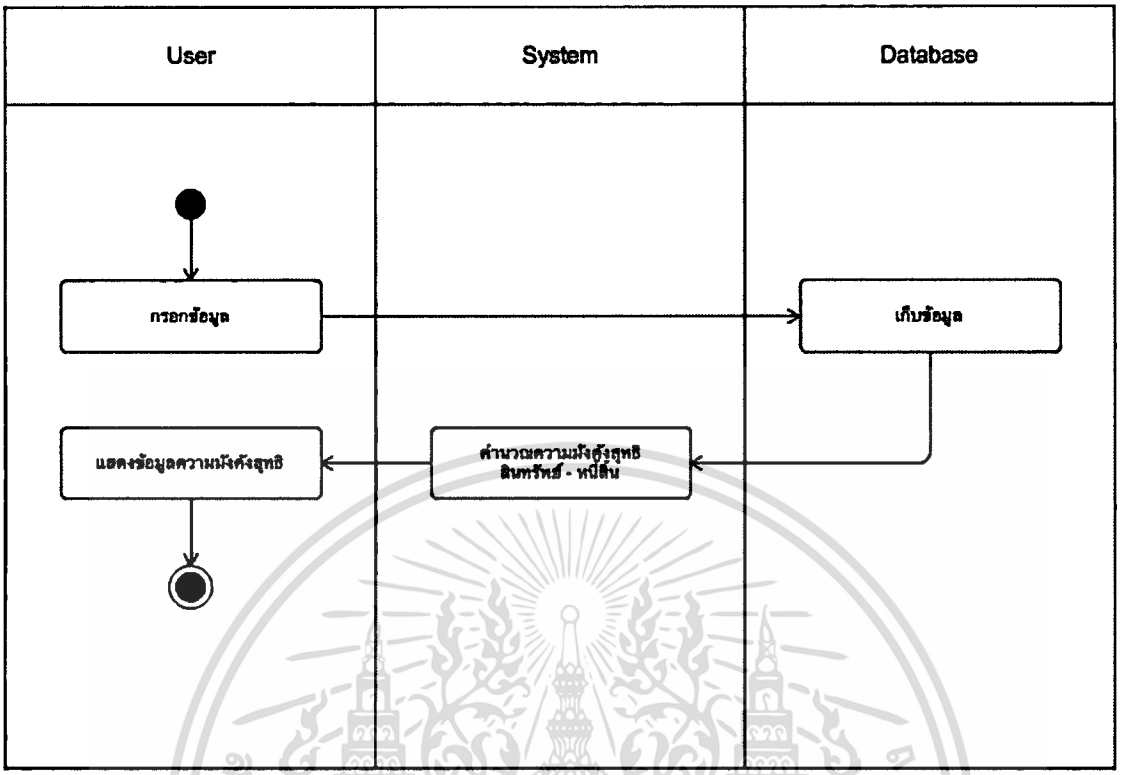
3.5.1 Realm Database

Realm เป็น Mobile Database มีจุดเด่นคือการใช้ Engine ในการจัดเก็บข้อมูลของตัวเองที่ออกแบบให้มีความเรียบง่าย (Simplicity) และเร็ว (Speed) ต่อไปนี้เป็นข้อดีของ Realm

1) Easy to Use ใช้งานง่าย ด้วยความที่ Realm data models เป็น Object ธรรมดาแบบ Object ของภาษา (Objective-c ,Swift) ซึ่งมันทำให้เราสามารถใช้งาน Data models ที่เราสร้างขึ้นมาเหมือนกับ Data models เดิมของเรา ไม่ว่าจะเป็น Subclass หรือ Properties

2) Cross-Platform Realm supports iOS & OS X (Objective-C & Swift) & Android. ซึ่งเราสามารถ เอาไฟล์ Realm ไปใช้กับ ทั้งสอง Platforms ได้เลย เหมาะมากสำหรับใครที่ต้องการทำ แอพทั้ง iOS และ Android

ส่วนที่ 5 คือ Activity Diagram “ตรวจสุขภาพการเงิน”



รูป 3.11 Activity Diagram Finance Health

3.5 ส่วนการเก็บฐานข้อมูล

ส่วนการเก็บฐานข้อมูลจะกล่าวถึงองค์ประกอบของส่วนต่อประสานกับผู้ใช้ และการส่งข้อมูลระหว่างส่วนต่อประสานกับผู้ใช้บนไอโอเอส (iOS) และฐานข้อมูลเรียล (Realm Database)

3.5.1 Realm Database

Realm เป็น Mobile Database มีจุดเด่นคือการใช้ Engine ในการจัดเก็บข้อมูลของตัวเอง ที่ออกแบบให้มีความเรียบง่าย (Simplicity) และเร็ว (Speed) ต่อไปนี้เป็นข้อดีของ Realm

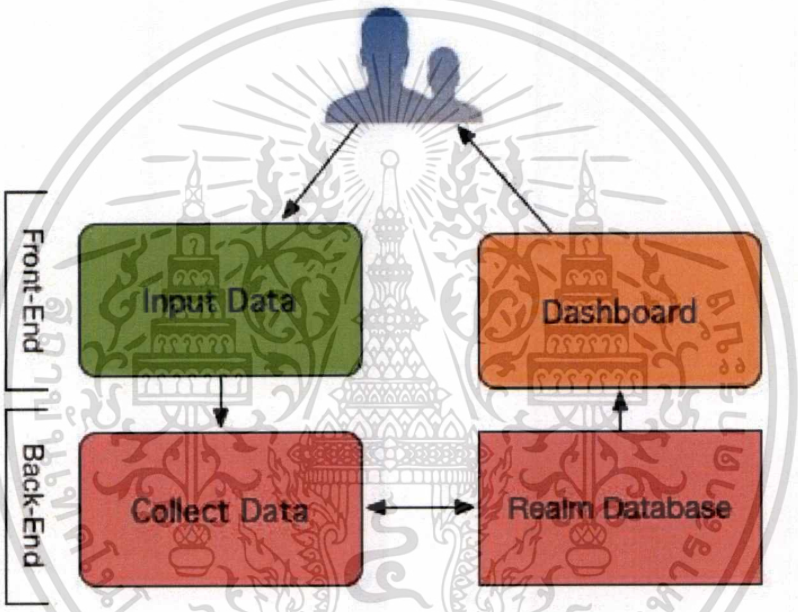
1) Easy to Use ใช้งานง่าย ด้วยความที่ Realm data models เป็น Object ธรรมดาแบบ Object ของภาษา (Objective-c ,Swift) ซึ่งมันทำให้เราสามารถใช้งาน Data models ที่เราสร้างขึ้นมาเหมือนกับ Data models เดิมของเรา ไม่ว่าจะเป็น Subclass หรือ Properties

2) Cross-Platform Realm supports iOS & OS X (Objective-C & Swift) & Android. ซึ่งเราสามารถ เอาไฟล์ Realm ไปใช้กับ ทั้งสอง Platforms ได้เลย เหมาะมากสำหรับใครที่ต้องการทำ แอพทั้ง iOS และ Android

3) Advanced Realm supports advanced features เช่น Encryption, Graph queries และที่สำคัญมัน Migrations ง่ายกว่า Core Data

3.6 โครงสร้างโดยรวมของแอพพลิเคชั่น

โครงสร้างของซอฟต์แวร์ประกอบด้วย 2 ส่วน ดังนี้ ส่วนที่ 1 เป็นส่วนต่อประสานกับผู้ใช้ที่อธิบายองค์ประกอบของหน้าแรก หน้าฟังก์ชั่นต่างๆ หน้าแสดงผลลัพธ์ทั้งหมด และหน้าตั้งค่า และส่วนที่ 2 เป็นส่วนส่งข้อมูลระหว่างส่วนต่อประสานกับผู้ใช้และฐานข้อมูล ซึ่งอธิบายรายละเอียดการส่งข้อมูลติดต่อกับฐานข้อมูลด้วยภาษาเรียล (Realm) ในทุกหน้าของแอพพลิเคชั่นอย่างละเอียด

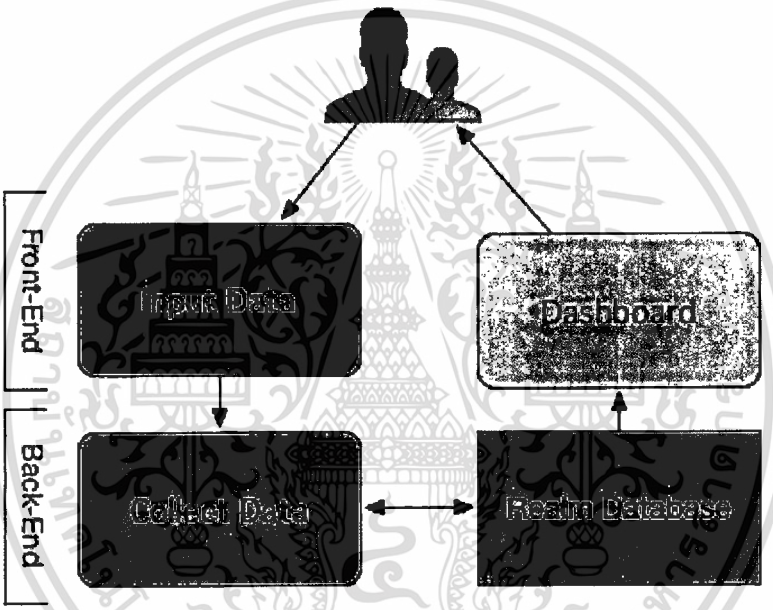


รูป 3.13 ภาพรวมแอพพลิเคชั่น

3) Advanced Realm supports advanced features เช่น Encryption, Graph queries และที่สำคัญมัน Migrations ง่ายกว่า Core Data

3.6 โครงสร้างโดยรวมของแอพพลิเคชั่น

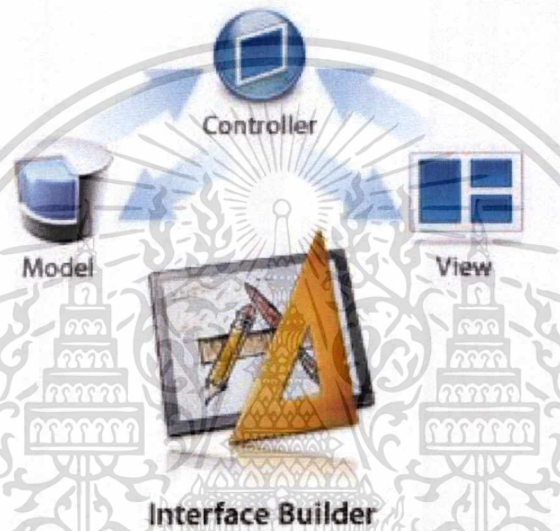
โครงสร้างของซอฟต์แวร์ประกอบด้วย 2 ส่วน ดังนี้ ส่วนที่ 1 เป็นส่วนต่อประสานกับผู้ใช้ที่อธิบายองค์ประกอบของหน้าแรก หน้าฟังก์ชั่นต่างๆ หน้าแสดงผลลัพธ์ทั้งหมด และหน้าตั้งค่า และส่วนที่ 2 เป็นส่วนส่งข้อมูลระหว่างส่วนต่อประสานกับผู้ใช้และฐานข้อมูล ซึ่งอธิบายรายละเอียดการส่งข้อมูลติดต่อกับฐานข้อมูลด้วยภาษาเรียล (Realm) ในทุกหน้าของแอพพลิเคชั่นอย่างละเอียด



รูป 3.13 ภาพรวมแอพพลิเคชั่น

3.7 โครงสร้างโปรแกรมที่พัฒนา

ก่อนที่จะเริ่มเขียน iOS Application ได้ก็ควรจะเริ่มจากสิ่งที่ผู้เริ่มต้นพัฒนาควรจะศึกษาพื้นฐานเริ่มต้นด้วยการแจกแจงองค์ประกอบของ MVC ออกมาก่อนเลย MVC เป็น Design Pattern ในระดับ Architecture ที่ได้รับความนิยมในการ Design Application ขนาดใหญ่ในปัจจุบัน ตัว MVC ย่อมาจากคำว่า Model-View-Controller ซึ่งก็ตรงตัวกับลักษณะของมัน ซึ่งก็คือ “การแบ่งองค์ประกอบของ Application ออกเป็นสามส่วนแยกออกจากกัน” ซึ่งแต่ละส่วนจะทำงานเชื่อมโยงกันในลักษณะดังรูป 3.14



รูป 3.14 Model View Controller

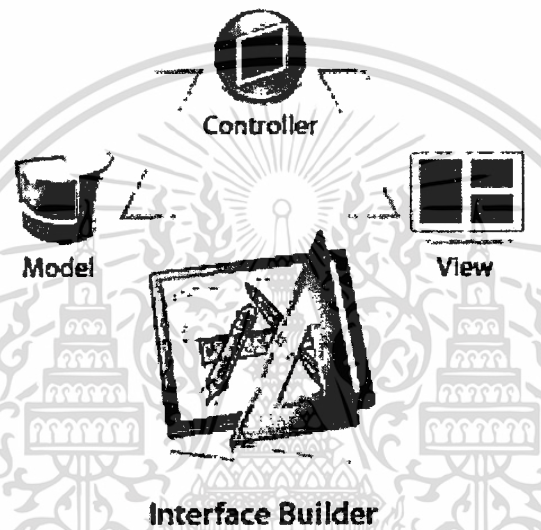
โดยหลักการแบ่งเป็นสามส่วนด้วยกันดังนี้

3.7.1 Model

ส่วนของข้อมูล ตัวอย่างเช่น Database หรือ Datasource ของตาราง Application เป็นต้น Model อาจรวมถึง State ของ Application ด้วยเช่นกัน ซึ่ง Model จะต้องมี Model Logic ซึ่งก็หมายถึง เงื่อนไขการเข้าถึงหรือการเก็บข้อมูล จะเก็บอย่างไร ชนิดไหน อย่างไร ฯลฯ หน้าที่ของ Model คือ “การเก็บข้อมูล” และ “การส่งข้อมูล”

3.7 โครงสร้างโปรแกรมที่พัฒนา

ก่อนที่จะเริ่มเขียน iOS Application ได้ก็ควรจะเริ่มจากสิ่งที่ผู้เริ่มต้นพัฒนาควรจะศึกษาพื้นฐานเริ่มต้นด้วยการแจกแจงองค์ประกอบของ MVC ออกมาก่อนเลย MVC เป็น Design Pattern ในระดับ Architecture ที่ได้รับความนิยมในการ Design Application ขนาดใหญ่ในปัจจุบัน ตัว MVC ย่อมาจากคำว่า Model-View-Controller ซึ่งก็ตรงตัวกับลักษณะของมัน ซึ่งก็คือ "การแบ่งองค์ประกอบของ Application ออกเป็นสามส่วนแยกออกจากกัน" ซึ่งแต่ละส่วนจะทำงานเชื่อมโยงกันในลักษณะดังรูป 3.14



รูป 3.14 Model View Controller

โดยหลักการแบ่งเป็นสามส่วนด้วยกันดังนี้

3.7.1 Model

ส่วนของข้อมูล ตัวอย่างเช่น Database หรือ Datasource ของตาราง Application เป็นต้น Model อาจรวมถึง State ของ Application ด้วยเช่นกัน ซึ่ง Model จะต้องมี Model Logic ซึ่งก็หมายถึง เงื่อนไขการเข้าถึงหรือการเก็บข้อมูล จะเก็บอย่างไร ชนิดไหน อย่างไร ฯลฯ หน้าที่ของ Model คือ "การเก็บข้อมูล" และ "การส่งข้อมูล"

3.7.2 View

ส่วนของการแสดงผล หน้าที่ของ View คือ "การแสดงผล" และ "การรอรับคำสั่ง" View ก็คล้างคลึงกับ GUI ซึ่งภายใน View ก็เหมือนกับ Model ต้องมี View Logic เช่นกัน View Logic ก็คือข้อกำหนดของการแสดงผล หน้าต่างกว้าง-ยาว, ปุ่มเป็นสไตล์ไหน, ตัวอักษรสีอะไร ฯลฯ

3.7.3 Controller

ส่วนนี้เป็นส่วนที่เชื่อมการทำงานระหว่าง Model กับ View ซึ่งก็คือ "User Logic หรือ Action Logic" เพราะเป็นส่วนควบคุมการทำงานของ User ที่ทำต่อ Application ว่า User ทำอะไรได้บ้าง แล้ว Application ต้องทำอะไรบ้างเมื่อได้รับ Action นั่นมา

3.8 โครงสร้างแอพพลิเคชั่นพัฒนา

โดยพัฒนาบนตัวโปรแกรม Xcode และเป็นเครื่องมือสำหรับนักพัฒนาโปรแกรม และแอพพลิเคชั่นบนแพลตฟอร์ม OS X และ iOS บนสมาร์ทโฟนที่เรารู้จักกันดีอย่างแอพพลิเคชั่นบน iPhone

สำหรับนักพัฒนาที่ต้องการจะพัฒนาแอพพลิเคชั่นบน iOS นั้นจำเป็นต้องมี XCode IDE ติดตั้งในเครื่องคอมพิวเตอร์ก่อน ซึ่งต่อไปนี้จะอธิบายถึงโครงสร้างทั้งหมดของโปรแกรม

รูป 3.15 โครงสร้างหลักแอพพลิเคชั่น

จากรูป 3.15 จะเห็นว่ามีโฟลเดอร์ต่างๆ ดังนี้

1) **Controller** โฟลเดอร์นี้เก็บไฟล์ที่ทำหน้าที่ให้ View และ Model ทำงานด้วยกันได้ หากเราเริ่มสร้าง New Project ขึ้นมาใหม่จาก Xcode เราจะได้ File ที่ Xcode สร้างให้โดยอัตโนมัติ ชื่อว่า AppDelegate และ ViewController

2) **iFin** คือชื่อแอพพลิเคชั่นที่ทำการพัฒนาขึ้น ในโฟลเดอร์นี้หลัก จะส่วนการทำงานของ

3.7.2 View

ส่วนของการแสดงผล หน้าที่ของ View คือ "การแสดงผล" และ "การรอรับคำสั่ง" View ก็คล้างคลึงกับ GUI ซึ่งภายใน View ก็เหมือนกับ Model ต้องมี View Logic เช่นกัน View Logic ก็คือข้อกำหนดของการแสดงผล หน้าต่างกว้าง-ยาว, ปุ่มเป็นสไตล์ไหน, ตัวอักษรสีอะไร ฯลฯ

3.7.3 Controller

ส่วนนี้เป็นส่วนที่เชื่อมการทำงานระหว่าง Model กับ View ซึ่งก็คือ "User Logic หรือ Action Logic" เพราะเป็นส่วนควบคุมการทำงานของ User ที่ทำต่อ Application ว่า User ทำอะไรได้บ้าง แล้ว Application ต้องทำอะไรบ้างเมื่อได้รับ Action นั่นมา

3.8 โครงสร้างแอพพลิเคชั่นพัฒนา

โดยพัฒนาบนตัวโปรแกรม Xcode และเป็นเครื่องมือสำหรับนักพัฒนาโปรแกรม และแอพพลิเคชั่นบนแพลตฟอร์ม OS X และ iOS บนสมาร์ทโฟนที่เรารู้จักกันดีอย่างแอพพลิเคชั่นบน iPhone

สำหรับนักพัฒนาที่ต้องการจะพัฒนาแอพพลิเคชั่นบน iOS นั้นจำเป็นต้องมี XCode IDE ติดตั้งในเครื่องคอมพิวเตอร์ก่อน ซึ่งต่อไปนี้จะอธิบายถึงโครงสร้างทั้งหมดของโปรแกรม

รูป 3.15 โครงสร้างหลักแอพพลิเคชั่น

จากรูป 3.15 จะเห็นว่ามีโฟลเดอร์ต่างๆ ดังนี้

1) **Controller** โฟลเดอร์นี้เก็บไฟล์ที่ทำหน้าที่ให้ View และ Model ทำงานด้วยกันได้ หากเราเริ่มสร้าง New Project ขึ้นมาใหม่จาก Xcode เราจะได้ File ที่ Xcode สร้างให้โดยอัตโนมัติ ชื่อว่า AppDelegate และ ViewController

2) **iFin** คือชื่อแอพพลิเคชั่นที่ทำการพัฒนาขึ้น ในโฟลเดอร์นี้หลัก จะส่วนการทำงานของ

แอพพลิเคชั่นเกือบทั้งหมดอยู่ในนี้

3) **Product** เป็นกลุ่มของแอพพลิเคชั่นที่จะถูกนำออกไปใช้งานหรือเรียกอีกอย่างว่า Target เราสามารถสร้าง Target ได้มากกว่าหนึ่ง Target ในหนึ่งโปรเจค เพื่อเป็นการประหยัดซอร์สโค้ด

บางไฟล์ที่จำเป็นต้องใช้ร่วมกัน วิธีการสร้าง Target ใหม่ทำได้โดย ไปที่หน้า Target คลิกขวา

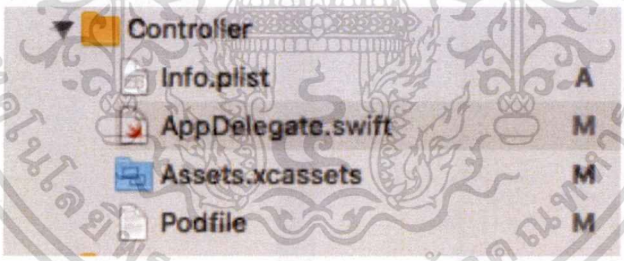
เลือก Duplicate Target

4) **Framework** เป็นกลุ่มของไลบรารี่ที่เราต้องนำเข้ามาเพื่อใช้ฟังก์ชั่นที่จำเป็น ซึ่งเฟรมเวิร์คเริ่มต้นที่จำเป็นสำหรับการทำแอพพลิเคชั่นมี 3 เฟรมเวิร์ค ได้แก่ UIKit มีฟังก์ชั่นเกี่ยวกับการ

สร้าง User Interface หลัก ของการพัฒนาแอพพลิเคชั่นบน iPhone/iPad ชื่อคลาสจะขึ้นต้นด้วย UI, Foundation มีฟังก์ชั่นพื้นฐานในการเขียนโปรแกรม และ CoreGraphics มีฟังก์ชั่นเกี่ยวกับการใช้งานกราฟฟิค ชื่อคลาสจะขึ้นต้นด้วย CG เราสามารถเพิ่ม Frameworks ใหม่ได้ที่ Build Phases ใน Target

5) **Pods** เป็นตัวจัดการระบบไลบรารี่ในแอพพลิเคชั่น

3.8.1 Controller Directory



รูป 3.16 Controller Directory

1) **Info.plist** คือ ไฟล์ในการตั้งค่าที่เกี่ยวข้องกับแอพพลิเคชั่น เช่น ชื่อแอพพลิเคชั่น เวอร์ชั่น หรือ nib ไฟล์เริ่มต้น อีกทั้งเรายังสามารถเพิ่มรายการใน Info.plist ได้ด้วยการคลิกขวา add row เลือกรายการที่ต้องการ เช่นการเพิ่มให้แอพพลิเคชั่นสามารถใช้WiFi ได้โดยเลือก Application uses wifi เป็นต้น

2) **AppDelegate** เป็นไฟล์ที่ควบคุมวัฏจักรของแอพพลิเคชั่น เช่นการเริ่มรันแอพพลิเคชั่น หยุดการทำงานเมื่อเกิดการขัดขวางการทำงานเช่น มีสายโทรเข้า หรือการกดปุ่ม Home การกลับสู่แอพพลิเคชั่น การปิดแอพพลิเคชั่น การรันในโหมด

แอพพลิเคชั่นเกือบทั้งหมดอยู่ในนี้

3) **Product** เป็นกลุ่มของแอพพลิเคชั่นที่จะถูกนำออกไปใช้งานหรือเรียกอีกอย่างว่า Target เรา สามารถสร้าง Target ได้มากกว่าหนึ่ง Target ในหนึ่งโปรเจค เพื่อเป็นการประหยัดซอร์สโค้ด

บางไฟล์ที่จำเป็นต้องใช้ร่วมกัน วิธีการสร้าง Target ใหม่ทำได้โดย ไปที่หน้า Target คลิกขวา

เลือก Duplicate Target

4) **Framework** เป็นกลุ่มของไลบรารี่ที่เราต้องนำเข้ามาเพื่อใช้ฟังก์ชั่นที่จำเป็น ซึ่งเฟรมเวิร์คเริ่มต้นที่จำเป็นสำหรับการทำแอพพลิเคชั่นมี 3 เฟรมเวิร์ค ได้แก่ UIKit มีฟังก์ชั่นเกี่ยวกับการ

สร้าง User Interface หลัก ของการพัฒนาแอพพลิเคชั่นบน iPhone/iPad ชื่อคลาสจะขึ้นต้นด้วย UI, Foundation มีฟังก์ชั่นพื้นฐานในการเขียนโปรแกรม และ CoreGraphics มีฟังก์ชั่นเกี่ยวกับการใช้งานกราฟฟิค ชื่อคลาสจะขึ้นต้นด้วย CG เราสามารถเพิ่ม Frameworks ใหม่ได้ที่ Build Phases ใน Target

5) **Pods** เป็นตัวจัดการระบบไลบรารี่ในแอพพลิเคชั่น

3.8.1 Controller Directory

รูป 3.16 Controller Directory

1) **Info.plist** คือ ไฟล์ในการตั้งค่าที่เกี่ยวข้องกับแอพพลิเคชั่น เช่น ชื่อแอพพลิเคชั่น เวอร์ชั่น หรือ nib ไฟล์เริ่มต้น อีกทั้งเรายังสามารถเพิ่มรายการใน Info.plist ได้ด้วยการคลิกขวา add row เลือกรายการที่ต้องการ เช่นการเพิ่มให้แอพพลิเคชั่นสามารถใช้WiFi ได้โดยเลือก Application uses wifi เป็นต้น

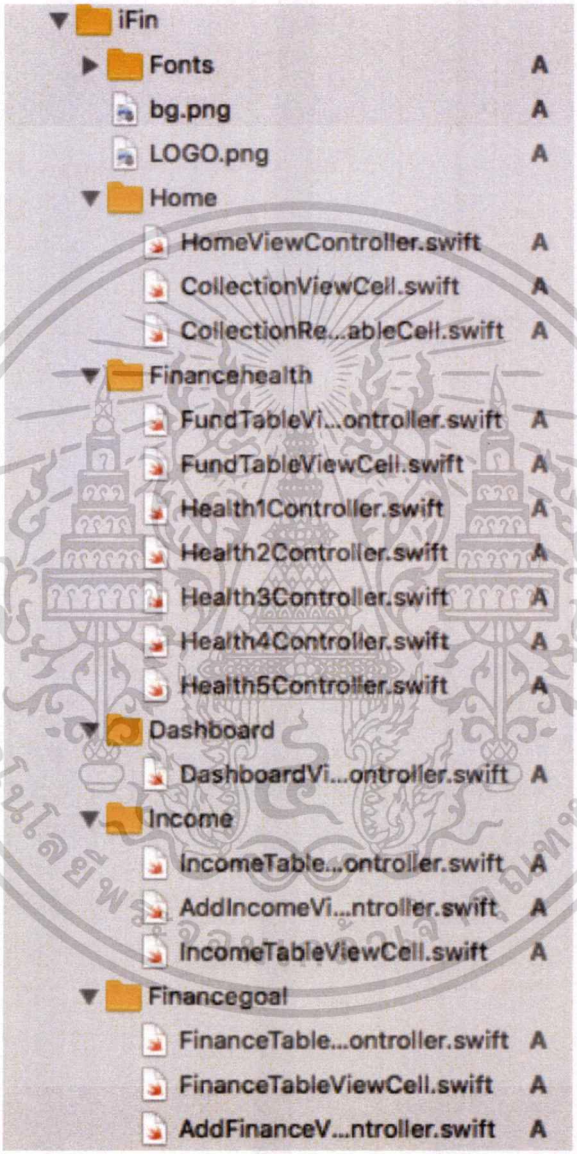
2) **AppDelegate** เป็นไฟล์ที่ควบคุมวัฏจักรของแอพพลิเคชั่น เช่นการเริ่มรันแอพพลิเคชั่น หยุดการทำงานเมื่อเกิดการขัดขวางการทำงานเช่น มีสายโทรเข้า หรือการกดปุ่ม Home การกลับสู่แอพพลิเคชั่น การปิดแอพพลิเคชั่น การรันในโหมด

Background ให้โปรแกรมยังทำงานอยู่ถึงแม้ผู้ใช้ปิดโปรแกรมไป

3) **Assets.xcassests** จัดการเกี่ยวกับ App icon ของเรา โดยเราสามารถเลือก Check ได้ว่า Application ของเรานั้น Support Device และ Model รุ่นไหนบ้าง

4) **Podfiles** คือไฟล์ไว้รันในส่วนของการติดตั้ง Library

3.8.2 iFin Directory



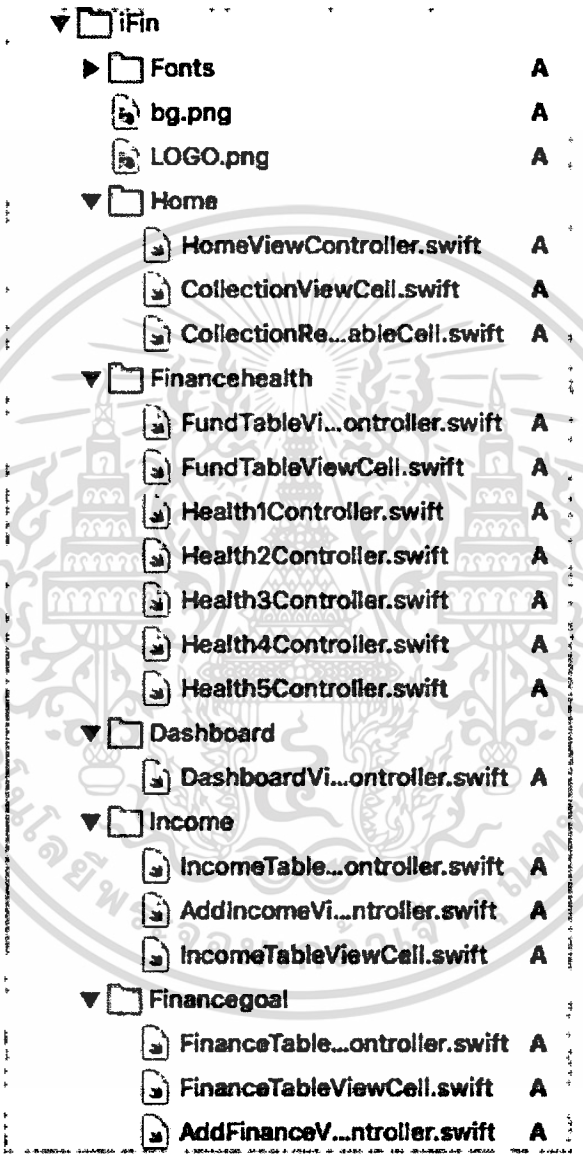
รูป 3.17 iFin Directory

Background ให้โปรแกรมยังทำงานอยู่ถึงแม้ผู้ใช้ปิดโปรแกรมไป

3) **Assets.xcassests** จัดการเกี่ยวกับ App icon ของเรา โดยเราสามารถเลือก Check ได้ว่า Application ของเรานั้น Support Device และ Model รุ่นไหนบ้าง

4) **Podfiles** คือไฟล์ไว้รันในส่วนของการติดตั้ง Library

3.8.2 iFin Directory



รูป 3.17 iFin Directory

1) **Fonts** เป็นส่วนเก็บ Custom Fonts ต่างๆในโปรแกรม ไว้สำหรับการตกแต่งส่วน UI ในการพัฒนา

2) **Home** เป็น Controller จัดการหน้าหลักบนแอพพลิเคชั่น โดยส่วนนี้ใช้การทำงาน UICollectionView ในการออกแบบ Collection View คือ จะอ่านข้อมูลที่อยู่ในรูปแบบของ Array ในรูปแบบต่าง ๆ และทำการ Loop แสดงรายการข้อมูลตาม Collection View Cell ที่ได้สร้างขึ้นมา โดยสามารถ Loop แสดงได้ทั้งแบบ แนวนอน และแนวตั้ง มี Module สำหรับการส่งค่าไปยัง Segue เพื่อเป็นการระบุ Identifier เมื่อทำการเปิดหน้าถัดไปในแอพพลิเคชั่น หลังจากหน้า Homeมีการส่ง Segue ไปทั้งหมด 3 ช่องโดยหน้าที่ต้องการส่ง ต้องกำหนด Identifier ตามหน้านี้ เพื่อที่ทำให้สามารถคลิกที่ช่องนี้แล้วบอกว่าให้ไปหน้าไหน

Identifier คือ TableChecklist1, TableChecklist2, TableChecklist3

3) **Financehealth** เป็น Controller จัดการหน้าตรวจสุขภาพการเงิน หลังจากคลิกเข้ามาผ่านหน้า Home โดยหลักในการออกแบบใช้ UITableView ในการออกแบบซึ่งจะแสดงเป็นตารางมี Module สำหรับการส่งค่าไปยัง Segue เพื่อเป็นการระบุ Identifier หากจะเปิดหน้าถัดไปในแอพพลิเคชั่น โดยหน้าที่ต้องการส่ง ต้อง
กำหนด Identifier ของแต่ละ Table โดยกำหนดไว้ดังนี้

Identifier คือ Health1, Health2, Health3, Health4, Health5

ในส่วนนี้จะมีการรับค่าอินพุทมาจาก ผู้ใช้ เมื่อหลังจากกรอกค่าอินพุทและกด Done เพื่อทำการเพิ่มค่าลง Database และแสดง Output บน View ภายในไฟล์ยังมี Controller เพิ่มเติมอีกคือส่วนที่มีการส่ง Segue จากหน้า Controller หลัก ไปยังหน้าถัดไป ภายในมี Module สำคัญมาโดยโมดุลที่ยกตัวอย่างมา ชื่อว่า prepareForSeugue เป็น Module ที่ไว้พาสค่าไปยังหน้าก่อนหน้านี้หลังจากการกด Done บนแอพพลิเคชั่นโดยส่วนนี้ได้รับค่าจาก TextField คือส่วนที่ผู้ใช้ได้กรอกค่าต่างๆลงบนแอพพลิเคชั่น

4) **Dashboard** เป็น Controller จัดการหน้าสรุป โดย Module หลักๆ จะมีสองส่วนคือส่วนของกราฟและส่วนของการคำนวณในการพาสดาต้าสำคัญๆ จากฟังก์ชั่นอื่นมาทำการคำนวณ ส่วนของกราฟจะเป็นการแสดงกราฟของรายได้ทั้งหมดของผู้ใช้ ว่าอยู่ ในเกณฑ์เท่าไร โดยใช้ Library ชื่อ iOS charts

1) **Fonts** เป็นส่วนเก็บ Custom Fonts ต่างๆในโปรแกรม ไว้สำหรับการตกแต่งส่วน UI ในการพัฒนา

2) **Home** เป็น Controller จัดการหน้าหลักบนแอพพลิเคชั่น โดยส่วนนี้ใช้การทำงาน UICollectionView ในการออกแบบ Collection View คือ จะอ่านข้อมูลที่อยู่ในรูปแบบของ Array ในรูปแบบต่าง ๆ และทำการ Loop แสดงรายการข้อมูลตาม Collection View Cell ที่ได้สร้างขึ้นมา โดยสามารถ Loop แสดงได้ทั้งแบบ แนวนอน และแนวตั้ง มี Module สำหรับการส่งค่าไปยัง Segue เพื่อเป็นการระบุ Identifier เมื่อทำการเปิดหน้าถัดไปในแอพพลิเคชั่น หลังจากหน้า Homeมีการส่ง Segue ไปทั้งหมด 3 ช่องโดยหน้าที่ต้องการส่ง ต้องกำหนด Identifier ตามหน้านี้ เพื่อที่ทำให้สามารถคลิกที่ช่องนี้แล้วบอกว่าให้ไปหน้าไหน

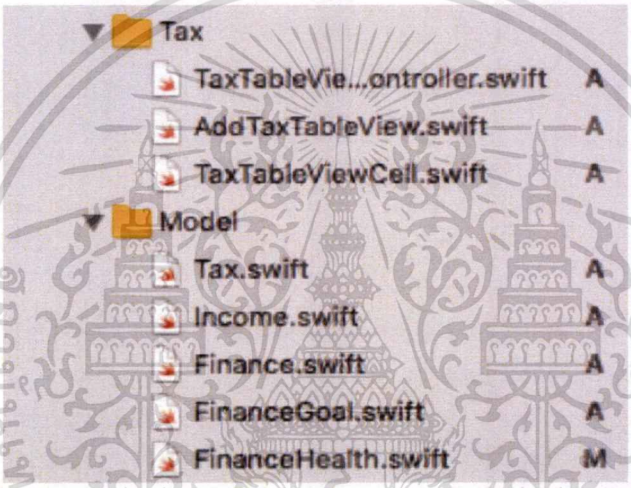
Identifier คือ TableChecklist1, TableChecklist2, TableChecklist3

3) **Financehealth** เป็น Controller จัดการหน้าตรวจสุขภาพการเงิน หลังจากคลิกเข้ามาผ่านหน้า Home โดยหลักในการออกแบบใช้ UITableView ในการออกแบบซึ่งจะแสดงเป็นตารางมี Module สำหรับการส่งค่าไปยัง Segue เพื่อเป็นการระบุ Identifier หากจะเปิดหน้าถัดไปในแอพพลิเคชั่น โดยหน้าที่ต้องการส่ง ต้อง
กำหนด Identifier ของแต่ละ Table โดยกำหนดไว้ดังนี้
Identifier คือ Health1, Health2, Health3, Health4, Health5
ในส่วนนี้จะมีการรับค่าอินพุทมาจาก ผู้ใช้ เมื่อหลังจากกรอกค่าอินพุทและกด Done เพื่อทำการเพิ่มค่าลง Database และแสดง Output บน View ภายในไฟล์ยังมี Controller เพิ่มเติมอีกคือส่วนที่มีการส่ง Segue จากหน้า Controller หลัก ไปยังหน้าถัดไป ภายในมี Module สำคัญมาโดยโมดุลที่ยกตัวอย่างมา ชื่อว่า prepareForSeugue เป็น Module ที่ไว้พาสค่าไปยังหน้าก่อนหน้านี้หลังจากการกด Done บนแอพพลิเคชั่นโดยส่วนนี้ได้รับค่าจาก TextField คือส่วนที่ผู้ใช้ได้กรอกค่าต่างๆลงบนแอพพลิเคชั่น

4) **Dashboard** เป็น Controller จัดการหน้าสรุป โดย Module หลักๆ จะมีสองส่วนคือส่วนของกราฟและส่วนของการคำนวณในการพาสดาต้าสำคัญๆ จากฟังก์ชั่นอื่นมาทำการคำนวณ ส่วนของกราฟจะเป็นการแสดงกราฟของรายได้ทั้งหมดของผู้ใช้ ว่าอยู่ ในเกณฑ์เท่าไร โดยใช้ Library ชื่อ iOS charts

5) **Income** เป็น Controller จัดการหน้าของรายได้ โดยหลักการออกแบบใช้ UITableView ซึ่งจะแสดงผลเป็นตารางให้ผู้ใช้ เข้ากรอกข้อมูลรายได้ต่างๆส่วนของ Module หลักๆ จะเป็นการเพิ่ม ลด จำนวนเงินเดือน ค่าเช่า ค่านายหน้า และ รายได้อื่นๆ เป็นหลัก หลังจากนั้น จะแสดง Output หลังจากผู้ใช้ได้กรอกอินพุตลงไป โดยแสดงเป็นรายได้รวม

6) **Finance Goal** เป็น Controller จัดการหน้าของเป้าหมายการเงิน โดยหลักการออกแบบใช้ UITableView ผู้ใช้ยังสามารถเพิ่ม ลบ ข้อมูลเป้าหมายการเงินได้ และส่วนของ Output ยังแสดงจำนวนเงินทั้งหมดที่ต้องใช้ และจำนวนเงินออมต่อเดือน เป็นต้น



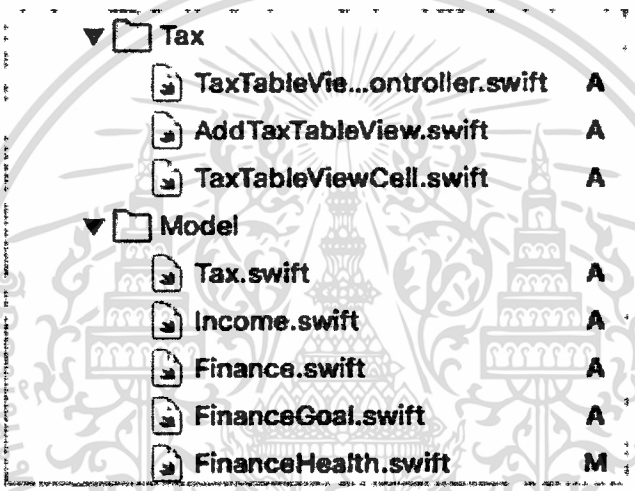
รูป 3.18 iFin Directory (ต่อ)

7) **Tax** เป็น Controller จัดการหน้าของภาษีส่วนบุคคลรายปี โดยคำนวณจาก รายได้ - รายจ่าย - จำนวนค่าลดหย่อน แสดงเป็นภาษีขั้นบันได และแสดงจำนวนภาษีที่ต้องจ่ายและมี จำนวนค่าลดหย่อนที่แสดงให้ผู้ใช้ได้เห็น โดย Module สำคัญหลักๆ จะเป็นหลักของการคำนวณ

8) **Model** เป็น Model ซึ่งเป็น Object ไว้เก็บข้อมูลต่างๆ ลง Database ซึ่งใช้ Database ของ Realm จำเป็นต้อง Import Realm เสียก่อน Realm Database ยังมีความหยืดหยุ่น ใช้งานง่าย สามารถเรียกใช้ได้ง่าย ซึ่งประกอบไปด้วย Model หลักในแอพพลิเคชั่นนี้มี Tax, Income, Finance, FinanceGoal, FinanceHealth เป็นต้น

5) **Income** เป็น Controller จัดการหน้าของรายได้ โดยหลักการออกแบบใช้ UITableView ซึ่งจะแสดงผลเป็นตารางให้ผู้ใช้ เข้ากรอกข้อมูลรายได้ต่างๆส่วนของ Module หลักๆ จะเป็นการเพิ่ม ลด จำนวนเงินเดือน ค่าเช่า ค่านายหน้า และ รายได้อื่นๆ เป็นหลัก หลังจากนั้น จะแสดง Output หลังจากผู้ใช้ได้กรอกอินพุตลงไป โดยแสดงเป็นรายได้รวม

6) **Finance Goal** เป็น Controller จัดการหน้าของเป้าหมายการเงิน โดยหลักการออกแบบใช้ UITableView ผู้ใช้ยังสามารถเพิ่ม ลบ ข้อมูลเป้าหมายการเงินได้ และส่วนของ Output ยังแสดงจำนวนเงินทั้งหมดที่ต้องใช้ และจำนวนเงินออมต่อเดือน เป็นต้น



รูป 3.18 iFin Directory (ต่อ)

7) **Tax** เป็น Controller จัดการหน้าของภาษีส่วนบุคคลรายปี โดยคำนวณจาก รายได้ - รายจ่าย - จำนวนค่าลดหย่อน แสดงเป็นภาษีขั้นบันได และแสดงจำนวนภาษีที่ต้องจ่ายและมี จำนวนค่าลดหย่อนที่แสดงให้ผู้ใช้ได้เห็น โดย Module สำคัญหลักๆ จะเป็นหลักของการคำนวณ

8) **Model** เป็น Model ซึ่งเป็น Object ไว้เก็บข้อมูลต่างๆ ลง Database ซึ่งใช้ Database ของ Realm จำเป็นต้อง Import Realm เสียก่อน Realm Database ยังมีความหยืดหยุ่น ใช้งานง่าย สามารถเรียกใช้ได้ง่าย ซึ่งประกอบไปด้วย Model หลักในแอพพลิเคชั่นนี้มี Tax, Income, Finance, FinanceGoal, FinanceHealth เป็นต้น

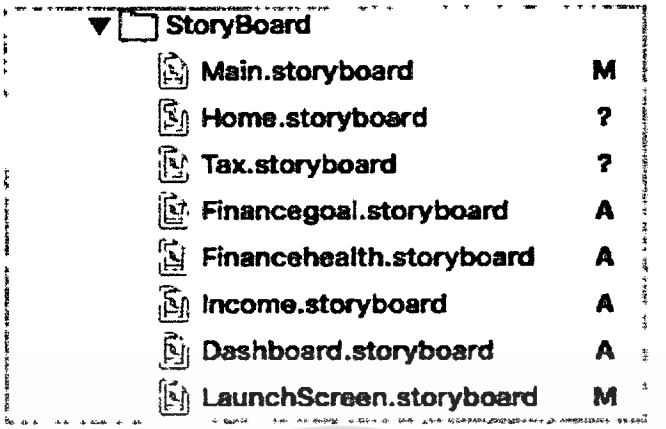
3.8.3 Storyboard Directory

รูป 3.19 Storyboard Directory

StoryBoard เป็น View ของหน้าตา UI ทั้งหมด ทั้งนี้เรายังสามารถปรับตั้งได้โดยไม่ต้องยุ่งกับส่วนของโปรแกรมมิ่งก็ทำได้ โดยหลักๆทำงานได้ดังนี้

1) สามารถมองภาพรวมของโปรแกรมได้ชัดเจนและสามารถมองเห็นความสัมพันธ์ของแต่ละ Interface ได้ ซึ่งโดยปกติแล้วต้องไปดูเอาเองใน Code
2) จัดการการเชื่อมต่อของ Interface ต่างๆได้ง่าย โดยไม่ต้องไปเรียกใช้งานผ่าน Method ในส่วนของ Code
3) รวมการจัดการเกี่ยวกับ User Interface ไว้ใน File เดียวซึ่งทำให้การทำงานสะดวกมาก จากปกติเราต้องไปเขียน Code สร้างความสัมพันธ์ระหว่าง Interface ในแต่ละ Object แต่ละตัว
4) ทำงานร่วมกันกับ Table View ได้เป็นอย่างดี สามารถจัดการได้ว่าในแต่ละ Cell นั้นมีความสัมพันธ์กับ Interface อื่นๆ อย่างไร
5) ลดการเขียน Code ลงไปได้มากอย่างเห็นได้ชัด

3.8.3 Storyboard Directory

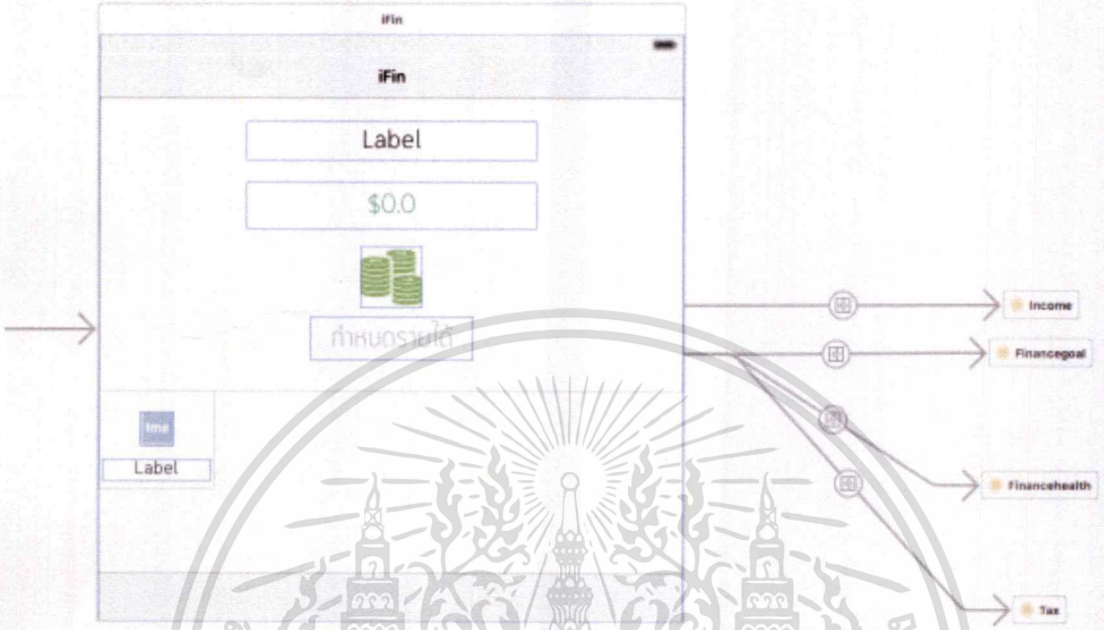


รูป 3.19 Storyboard Directory

StoryBoard เป็น View ของหน้าตา UI ทั้งหมด ทั้งนี้เรายังสามารถปรับตั้งได้โดยไม่ต้องยุ่งกับส่วนของโปรแกรมมิ่งก็ทำได้ โดยหลักๆทำงานได้ดังนี้

1) สามารถมองภาพรวมของโปรแกรมได้ชัดเจนและสามารถมองเห็นความสัมพันธ์ของแต่ละ Interface ได้ ซึ่งโดยปกติแล้วต้องไปดูเอาเองใน Code
2) จัดการการเชื่อมต่อของ Interface ต่างๆได้ง่าย โดยไม่ต้องไปเรียกใช้งานผ่าน Method ในส่วนของ Code
3) รวมการจัดการเกี่ยวกับ User Interface ไว้ใน File เดียวซึ่งทำให้การทำงานสะดวกมาก จากปกติเราต้องไปเขียน Code สร้างความสัมพันธ์ระหว่าง Interface ในแต่ละ Object แต่ละตัว
4) ทำงานร่วมกันกับ Table View ได้เป็นอย่างดี สามารถจัดการได้ว่าในแต่ละ Cell นั้นมีความสัมพันธ์กับ Interface อื่นๆ อย่างไร
5) ลดการเขียน Code ลงไปได้มากอย่างเห็นได้ชัด

ส่วนของด้านขวาของ View จะเป็นส่วน Segue ที่เชื่อมต่อไปยังหน้าต่อไป โดยแยก Storyboard ตามฟังก์ชั่นบนแอพพลิเคชั่นดังรูป 3.20



รูป 3.20 Storyboard (HomeView)

3.9 โครงสร้างหลักของโค้ด

3.9.1 โครงสร้างสำหรับ User Interface

1) **UICollectionView** คือ จะอ่านข้อมูลที่อยู่ในรูปแบบของ Array ในรูปแบบต่าง ๆ และทำการ Loop แสดงรายการข้อมูลตาม Collection View Cell ที่ได้สร้างขึ้นมา โดยสามารถ Loop แสดงได้ทั้งแบบ แนวนอน และแนวตั้ง

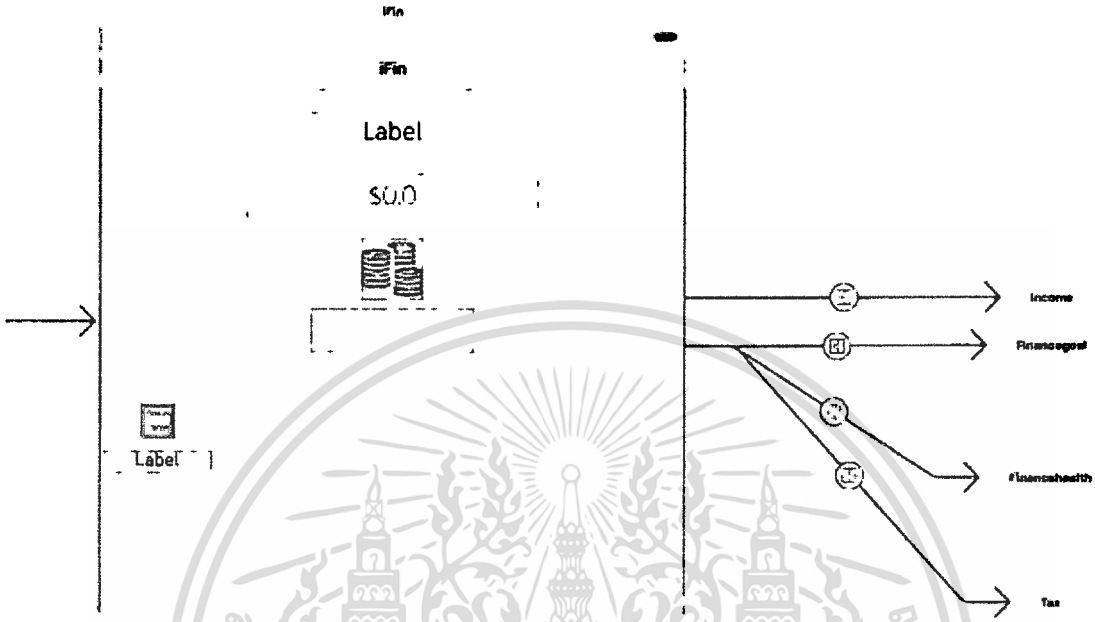
ฟังก์ชั่นสำคัญในการพัฒนามีดังนี้

Func collectionView numberOfItemsInSection คือ ฟังก์ชั่นที่รีเทิร์นจำนวน Section ทั้งหมดที่จะแสดงเป็น View

Func collectionView cellForItemAtIndexPath คือ ฟังก์ชั่นที่รีเทิร์นรายละเอียดบน CollectionView Cell หรือ จะให้แสดง Label หรืออื่นๆ เป็นต้น

Func collectionView didSelectItemAtIndexPath คือ ฟังก์ชั่นสามารถโปรแกรมมิ่งให้หลังจากที่เรากด Cell นั้นๆ เช่น กำหนดให้ Cell ที่ 0 มี Identifier ชื่อ ident1

ส่วนของด้านขวาของ View จะเป็นส่วน Segue ที่เชื่อมต่อไปยังหน้าต่อไป โดยแยก Storyboard ตามฟังก์ชั่นบนแอพพลิเคชั่นดังรูป 3.20



รูป 3.20 Storyboard (HomeView)

3.9 โครงสร้างหลักของโค้ด

3.9.1 โครงสร้างสำหรับ User Interface

1) **UICollectionView** คือ จะอ่านข้อมูลที่อยู่ในรูปแบบของ Array ในรูปแบบต่าง ๆ และทำการ Loop แสดงรายการข้อมูลตาม Collection View Cell ที่ได้สร้างขึ้นมา โดยสามารถ Loop แสดงได้ทั้งแบบ แนวนอน และแนวตั้ง

ฟังก์ชั่นสำคัญในการพัฒนามีดังนี้

Func collectionView numberOfItemsInSection คือ ฟังก์ชั่นที่รีเทิร์นจำนวน Section ทั้งหมดที่จะแสดงเป็น View

Func collectionView cellForItemAtIndexPath คือ ฟังก์ชั่นที่รีเทิร์นรายละเอียดบน CollectionView Cell หรือ จะให้แสดง Label หรืออื่นๆ เป็นต้น

Func collectionView didSelectItemAtIndexPath คือ ฟังก์ชั่นสามารถโปรแกรมมิ่งให้หลังจากที่เรากด Cell นั้นๆ เช่น กำหนดให้ Cell ที่ 0 มี Identifier ชื่อ ident1

Func prepareForSegue คือ ฟังก์ชั่นสำหรับการส่งค่าไปยัง หน้าอื่น โดยการพาสค่าไปยังตัวแปรที่สร้างขึ้น

2) **UITableview** หรือตาราง ตามชื่อ แต่สำหรับ UITableView ที่อยู่บน iOS จะมีข้อจำกัดคือมีแค่ 1 Column และมีได้หลาย Row หลาย Section แล้วสิ่งที่ Table ต้องมีคือจำนวน Row และข้อมูลที่อยู่ในแต่ละ Row โดยทั้งหมดนี้เราจะเรียกว่า Data Souce
ส่วน Delegate ถ้าแปลตามตัวจะหมายถึง ตัวแทน ตัวแทนในที่นี้ก็คือเป็นตัวที่ทำหน้าที่เมื่อมีการทำ Action อะไรกับ ข้อมูลใน Table ของเรา เช่นเมื่อ User เลือกที่ Row ใด Row หนึ่งของ Table เราก็จะรู้ได้แล้วว่ามี Action เกิดขึ้น จากนั้นเมื่อเราต้องการให้ทำ Action อะไรต่อ ก็จัดการได้ที่ส่วนของ Delegate ของ UITableView นี้
ฟังก์ชั่นสำคัญในการพัฒนามีดังนี้

Func tableView numberOfItemsInSection คือ ฟังก์ชั่นที่รีเทิร์นจำนวน Section ทั้งหมดที่จะแสดงเป็น View

Func tableView cellForItemAtIndexPath คือ ฟังก์ชั่นที่รีเทิร์นรายละเอียดบน TableView Cell หรือ จะให้แสดง Label หรืออื่นๆ เป็นต้น

Func tableView didSelectItemAtIndexPath คือ ฟังก์ชั่นสามารถโปรแกรมมิ่งให้หลังจากที่เรากด Cell นั้นๆ เช่น กำหนดให้ Cell ที่ 0 มี Identifier ชื่อ ident1

Func prepareForSegue คือ ฟังก์ชั่นสำหรับการส่งค่าไปยัง หน้าอื่น โดยการพาสค่าไปยังตัวแปรที่สร้างขึ้น

3) **UIViewController** ส่วนมากจะเกี่ยวข้องกับการแสดงผล Viewและการซ่อน/ปิด View ประกอบด้วย viewDidLoad, viewWillAppear, viewDidAppear, viewWillDisappear, viewDidDisappear ซึ่งเราจะมาทำความรู้จักกันว่าในแต่ละ Event นั้น ถูกเรียกใช้งานเมื่อไร Event ไหนถูกเรียกก่อนหรือหลัง
อีเวนท์ที่ถูกเรียกใช้โดยอัตโนมัติของ ViewController เมื่อเกิดเหตุการณ์ใดเหตุการณ์หนึ่งเกี่ยวกับการแสดงผล View หรือซ่อน/ปิด View
ฟังก์ชั่นสำคัญในการพัฒนามีดังนี้

Func viewDidLoad - Method นี้จะถูกเรียกใช้งานเมื่อเราทำการเรียกเปิด View ครั้งแรก หรือพูดง่าย ๆ คือเราเริ่มสั่ง load view มันก็จะเข้ามาทำงานที่ method นี้ก่อนเลย เพื่อให้เราสามารถที่จะเริ่ม Initialize Code ต่าง ๆ หรือกำหนดค่าเริ่มต้นต่าง ๆ เกี่ยวกับ View

Func prepareForSegue คือ ฟังก์ชั่นสำหรับการส่งค่าไปยัง หน้าอื่น โดยการพาสค่าไปยังตัวแปรที่สร้างขึ้น

2) **UITableview** หรือตาราง ตามชื่อ แต่สำหรับ UITableView ที่อยู่บน iOS จะมีข้อจำกัดคือมีแค่ 1 Column และมีได้หลาย Row หลาย Section แล้วสิ่งที่ Table ต้องมีคือจำนวน Row และข้อมูลที่อยู่ในแต่ละ Row โดยทั้งหมดนี้เราจะเรียกว่า Data Souce
ส่วน Delegate ถ้าแปลตามตัวจะหมายถึง ตัวแทน ตัวแทนในที่นี้ก็คือเป็นตัวที่ทำหน้าที่เมื่อมีการทำ Action อะไรกับ ข้อมูลใน Table ของเรา เช่นเมื่อ User เลือกที่ Row ใด Row หนึ่งของ Table เราก็จะรู้ได้แล้วว่ามี Action เกิดขึ้น จากนั้นเมื่อเราต้องการให้ทำ Action อะไรต่อ ก็จัดการได้ที่ส่วนของ Delegate ของ UITableView นี้
ฟังก์ชั่นสำคัญในการพัฒนามีดังนี้

Func tableView numberOfItemsInSection คือ ฟังก์ชั่นที่รีเทิร์นจำนวน Section ทั้งหมดที่จะแสดงเป็น View

Func tableView cellForItemAtIndexPath คือ ฟังก์ชั่นที่รีเทิร์นรายละเอียดบน TableView Cell หรือ จะให้แสดง Label หรืออื่นๆ เป็นต้น

Func tableView didSelectItemAtIndexPath คือ ฟังก์ชั่นสามารถโปรแกรมมิ่งให้หลังจากที่เรากด Cell นั้นๆ เช่น กำหนดให้ Cell ที่ 0 มี Identifier ชื่อ ident1

Func prepareForSegue คือ ฟังก์ชั่นสำหรับการส่งค่าไปยัง หน้าอื่น โดยการพาสค่าไปยังตัวแปรที่สร้างขึ้น

3) **UIViewController** ส่วนมากจะเกี่ยวข้องกับการแสดงผล Viewและการซ่อน/ปิด View ประกอบด้วย viewDidLoad, viewWillAppear, viewDidAppear, viewWillDisappear, viewDidDisappear ซึ่งเราจะมาทำความรู้จักกันว่าในแต่ละ Event นั้น ถูกเรียกใช้งานเมื่อไร Event ไหนถูกเรียกก่อนหรือหลัง
อีเวนท์ที่ถูกเรียกใช้โดยอัตโนมัติของ ViewController เมื่อเกิดเหตุการณ์ใดเหตุการณ์หนึ่งเกี่ยวกับการแสดงผล View หรือซ่อน/ปิด View
ฟังก์ชั่นสำคัญในการพัฒนามีดังนี้

Func viewDidLoad - Method นี้จะถูกเรียกใช้งานเมื่อเราทำการเรียกเปิด View ครั้งแรก หรือพูดง่าย ๆ คือเราเริ่มสั่ง load view มันก็จะเข้ามาทำงานที่ method นี้ก่อนเลย เพื่อให้เราสามารถที่จะเริ่ม Initialize Code ต่าง ๆ หรือกำหนดค่าเริ่มต้นต่าง ๆ เกี่ยวกับ View

Func viewWillAppear - หลังจากที่ method "viewDidLoad" ถูกเรียกใช้งานเพื่อ load view เรียบร้อยแล้ว ตัว ViewController ก็จะเริ่มทำการแสดงผล view บนหน้าจอ ซึ่งก็คือ method นี้นั่นเอง (แปลชื่อ method ตรงตัวก็คือ View กำลังจะแสดงผลบนหน้าจอ)

Func viewDidAppear - เมื่อ View ถูกแสดงผลบนหน้าจอเรียบร้อยแล้ว method นี้จะถูกเรียกใช้งานต่อทันที ดังนั้นหากเราต้องการสั่ง code อะไรก็ตาม หลังจากที่หน้าจอแสดงผลแล้ว สามารถทำได้ที่ method นี้

Func viewWillDisappear - ความหมายคล้ายกับ "viewWillAppear" แต่จะตรงข้ามกันตรงที่ มันจะถูกทำงานเมื่อ View กำลังจะถูกซ่อนหรือปิดการแสดงผล

Func viewDidDisappear - ความหมายคล้ายกับ "viewDidAppear" แต่จะตรงข้ามกันตรงที่ มันจะถูกทำงานเมื่อ View นั้นถูกซ่อนหรือปิดการแสดงผลไปเรียบร้อยแล้ว

3.9.2 โครงสร้างหลักในการคำนวณ

1) โค้ดคำนวณในส่วนของ “ตรวจสุขภาพการเงิน” ยกตัวอย่างมาบางส่วน ดังนี้

```
let add = FinanceHealth()

if let check = sinsuball{
    var new : Double!
    let id = health!.id
    let up = arrayHealth[id-1]

    let sum = seg1! + seg2! + seg3! + seg4! + seg5!
    if let check1 = sinsublongtunall{
        if check1 != 0{
        new = new + sum
        }
        if let c = sinsublongprivateall{
        new = new + check1 + c
        }else{
            new = new + check1
        }
        realm.beginWrite()
        up.sinsubsapab = sum
        up.sinsuball = new
        realm.add(up,update: true)
        try! realm.commitWrite()
```

รูป 3.21 โค้ดคำนวณ “ตรวจสุขภาพการเงิน”

คำอธิบายโค้ดรูป 3.21

1.1) สร้างตัวแปรในการเก็บค่ามีชนิดเป็น FinanceHealth() หรือ Model หลังจากนั้นทำการเทียบค่า โดยใช้คำสั่ง if let หากตัวแปรชื่อ “sinsuball” ไม่เป็น nil

Func viewWillAppear - หลังจากที่ method "viewDidLoad" ถูกเรียกใช้งานเพื่อ load view เรียบร้อยแล้ว ตัว ViewController ก็จะเริ่มทำการแสดงผล view บนหน้าจอ ซึ่งก็คือ method นี้นั่นเอง (แปลชื่อ method ตรงตัวก็คือ View กำลังจะแสดงผลบนหน้าจอ)

Func viewDidAppear - เมื่อ View ถูกแสดงผลบนหน้าจอเรียบร้อยแล้ว method นี้จะถูกเรียกใช้งานต่อทันที ดังนั้นหากเราต้องการสั่ง code อะไรก็ตาม หลังจากที่หน้าจอแสดงผลแล้ว สามารถทำได้ที่ method นี้

Func viewWillDisappear - ความหมายคล้ายกับ "viewWillAppear" แต่จะตรงข้ามกันตรงที่ มันจะถูกทำงานเมื่อ View กำลังจะถูกซ่อนหรือปิดการแสดงผล

Func viewDidDisappear - ความหมายคล้ายกับ "viewDidAppear" แต่จะตรงข้ามกันตรงที่ มันจะถูกทำงานเมื่อ View นั้นถูกซ่อนหรือปิดการแสดงผลไปเรียบร้อยแล้ว

3.9.2 โครงสร้างหลักในการคำนวณ

1) โค้ดคำนวณในส่วนของ “ตรวจสุขภาพการเงิน” ยกตัวอย่างมาบางส่วน ดังนี้

```
add = FinanceHealth()

check = sinsuball{
    new : Double!
    id = health!.id
    up = arrayHealth[id-1]

    sum = seg1! + seg2! + seg3! + seg4! + seg5!
    check1 = sinsublongtunall{
    check1 != 0{
    new = new + sum
    }
    c = sinsublongprivateall{
    new = new + check1 + c
    }{
        new = new + check1
    }
    realm.beginWrite()
    up.sinsubsanab = sum
    up.sinsuball = new
    realm.add(up,update:
    realm.commitWrite()
```

รูป 3.21 โค้ดคำนวณ “ตรวจสุขภาพการเงิน”

คำอธิบายโค้ดรูป 3.21

1.1) สร้างตัวแปรในการเก็บค่ามีชนิดเป็น FinanceHealth() หรือ Model หลังจากนั้นทำการเทียบค่า โดยใช้คำสั่ง if let หากตัวแปรชื่อ “sinsuball” ไม่เป็น nil

1.2) สร้างตัวแปรเก็บค่าชนิด Double ชื่อ new และสร้างตัวแปรในการเรียก Object ที่มี Array ลำดับเดียวกับที่ต้องการแก้ไข ชื่อ up โดย id ให้อ้างอิงจาก index ช่องที่เราจะแก้ไข

1.3) สร้างตัวแปรในการเก็บค่า ที่ทำส่งมาจาก input ของผู้ใช้ที่กรอกลงบนแอพพลิเคชั่น โดยใช้คำสั่ง if let หากตัวแปรชื่อ “sinsublongtunall” ไม่เป็น nil

1.4) ภายในลูปจะเริ่มทำการเช็คค่าตามลำดับ หลังจากนั้นทำการ assign ค่าลงในตัวแปร

ที่มีชนิดเป็น Model ที่จะทำการ add หลังจากนั้นใช้คำสั่งในการ add

2) โค้ดคำนวณในส่วนของ “สรุป” ยกตัวอย่างมาบางส่วน ดังนี้

```
// tax
if arraytaxs != nil{
    // sitlodyorn
    sitlodyorn.text = arraytaxs!.formatUsingAbbrevation()
    print(arraytaxs!)
    // tax1
    var sumtax = arraysalary! * 12 - arraytaxs! - expenses - 30000
    print("\(arraysalary! * 12)")
    print(arraytaxs!)
    print(expenses)
    print(sumtax)
    if sumtax > 150000 && sumtax <= 300000 {
        sumtax = sumtax - 150000
        sumtax = sumtax * 0.05
        tax1.text = formatter.stringFromNumber(0.05)
    }else if sumtax == 0{
        sumtax = 0.0
        tax1.text = formatter.stringFromNumber(0)
    }else if sumtax >= 300001 && sumtax <= 500000{
        sumtax = sumtax - 300000
        sumtax = sumtax * 0.10
        sumtax = sumtax + sum1
        tax1.text = formatter.stringFromNumber(0.1)
    }else if sumtax >= 500001 && sumtax <= 750000{
        let sum1 = 150000 * 0.05
        let sum2 = 200000 * 0.10
        sumtax = sumtax - 500000
        sumtax = sumtax * 0.15
        sumtax = sumtax + sum1 + sum2
        tax1.text = formatter.stringFromNumber(0.15)
```

รูป 3.22 โค้ดคำนวณ “สรุป” ส่วนแสดง “ภาษี”

โค้ดคำอธิบายรูป 3.22 (ส่วนแสดงภาษี)

2.1.1) ในส่วนแรก ใช้คำสั่ง if เช็คตัวแปร arraytaxs ที่เก็บค่าใน Database (Model) Tax ทั้งหมด ถ้าไม่เป็น nil ให้เข้าลูป หลังจากนั้นดึง สร้างตัวแปรเก็บ sumtax ที่จะแสดงจำนวนภาษีที่ต้องจ่าย โดยหาจาก ตัวแปร arraysalaray ที่เก็บค่าเงินเดือนมา คำนวณเป็นต่อปี - ภาษีที่ต้องจ่าย - ค่าใช้จ่าย - ลดหย่อนเบื้องต้น 30,000 บาท

1.2) สร้างตัวแปรเก็บค่าชนิด Double ชื่อ new และสร้างตัวแปรในการเรียก Object ที่มี Array ลำดับเดียวกับที่ต้องการแก้ไข ชื่อ up โดย id ให้อ้างอิงจาก index ช่องที่เราจะแก้ไข

1.3) สร้างตัวแปรในการเก็บค่า ที่ทำส่งมาจาก input ของผู้ใช้ที่กรอกลงบนแอพพลิเคชั่น โดยใช้คำสั่ง if let หากตัวแปรชื่อ “sinsublongtunall” ไม่เป็น nil

1.4) ภายในลูปจะเริ่มทำการเช็คค่าตามลำดับ หลังจากนั้นทำการ assign ค่าลงในตัวแปร

ที่มีชนิดเป็น Model ที่จะทำการ add หลังจากนั้นใช้คำสั่งในการ add

2) โค้ดคำนวณในส่วนของ “สรุป” ยกตัวอย่างมาบางส่วน ดังนี้

รูป 3.22 โค้ดคำนวณ “สรุป” ส่วนแสดง “ภาษี”

โค้ดคำอธิบายรูป 3.22 (ส่วนแสดงภาษี)

2.1.1) ในส่วนแรก ใช้คำสั่ง if เช็คตัวแปร arraytaxs ที่เก็บค่าใน Database (Model) Tax ทั้งหมด ถ้าไม่เป็น nil ให้เข้าลูป หลังจากนั้นดึง สร้างตัวแปรเก็บ sumtax ที่จะแสดงจำนวนภาษีที่ต้องจ่าย โดยหาจาก ตัวแปร arraysalaray ที่เก็บค่าเงินเดือนมา คำนวณเป็นต่อปี - ภาษีที่ต้องจ่าย - ค่าใช้จ่าย - ลดหย่อนเบื้องต้น 30,000 บาท

2.1.2) หลังจากนั้นเช็คลูป if ตามภาษีขั้นบันได โดยแสดง output เป็น %

```
//income
if arrayincomes != nil{
    // income all
    if arraycommission != nil && arrayrent != nil && arrayother != nil{
        sumin = arraysalary! * 12 + arraycommission! + arrayrent! + arrayother!
    }else{
        sumin = arraysalary! * 12
    }
incomeAll.text = sumin.formatUsingAbbrevation()
    // expense
    expenses = sumin * 0.4
    if expenses > 60000{
        expenses = 60000
    }
    expense.text = expenses.formatUsingAbbrevation()
    // lodyorn
    if arraytaxs != nil {
         sum1 = 30000 + arraytaxs!
    lodyorn.text = sum1.formatUsingAbbrevation()
    }else{
        sum1 = 30000
        let lodyorn1 = 30000.0
        lodyorn.text = lodyorn1.formatUsingAbbrevation()
    }
    // sutti
    let suttisum = sumin - expenses - sum1
    sutti.text = suttisum.formatUsingAbbrevation()
}else{
    incomeAll.text = "0"
    // lodyorn
    let lodyorn1 = 30000.0
    lodyorn.text = lodyorn1.formatUsingAbbrevation()
    // expense
    expense.text = "0"
    // sutti
    sutti.text = "0"
}
```

รูป 3.23 โค้ดคำนวณ “สรุป” ส่วนแสดง “รายได้”

โค้ดคำอธิบายรูป 3.23 (ส่วนแสดงรายได้)

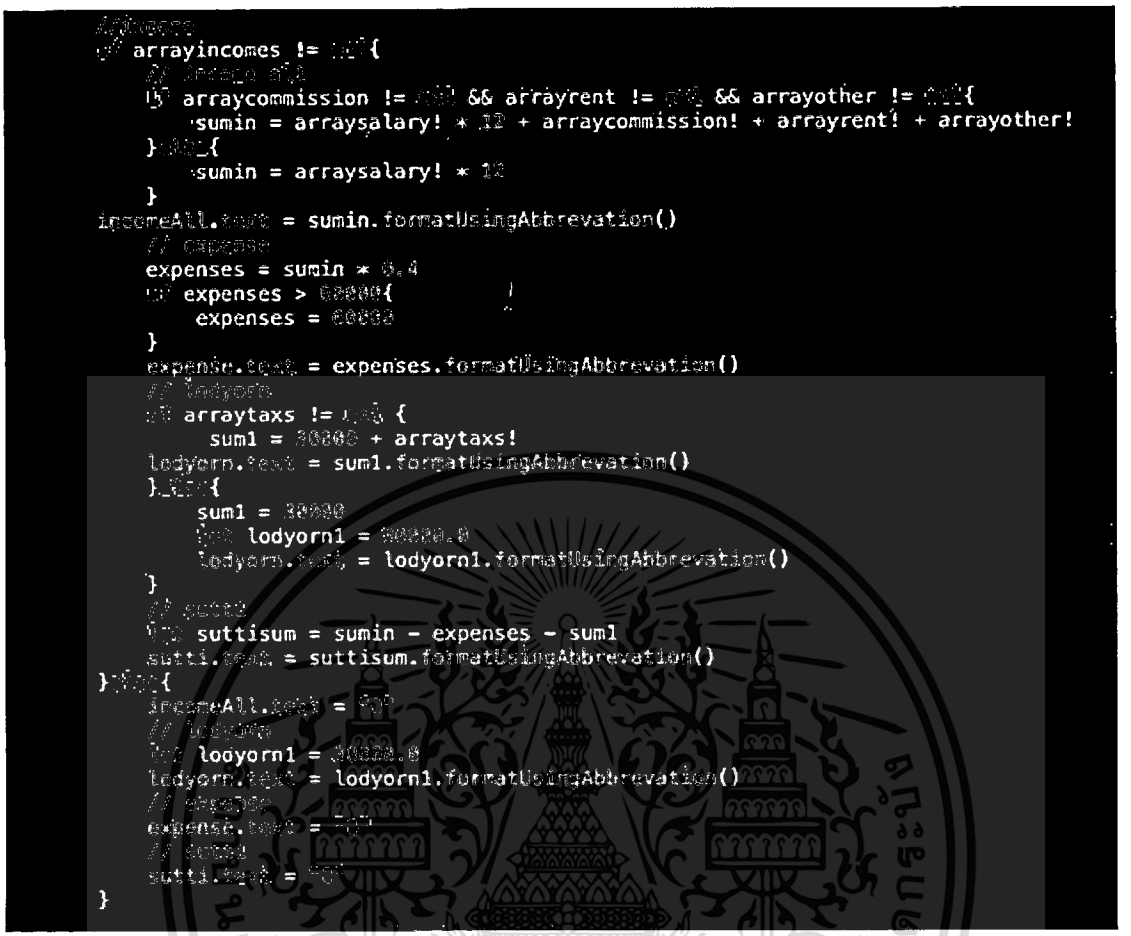
2.2.1) ในส่วนแรก ใช้คำสั่ง if เช็คตัวแปร arrayincomes ที่เก็บค่าใน Database (Model) Income ทั้งหมด ถ้าไม่เป็น nil ให้เข้าลูป

2.2.2) หลังจากนั้นเช็คลูป if โดยเช็ค ถ้าค่าเป็น nil ให้เข้าลูป if หรือ else ตามcondition นั้นๆ หลังจากนั้นแสดง output โดยดึงค่าจากตัวแปร sumin

2.2.3) ต่อมาเป็นส่วนแสดง รายจ่าย โดยคิดจาก sumin * 40% หาก output เกิน 60,000 บาทให้ตัวแปร expense = 60000 และแสดง outputโดยดึงค่าตัวแปรจาก expense และส่วนแสดง ลดหย่อนและเงินได้สุทธิ หลังจากคำนวณตามทฤษฎี ให้แสดง output โดยดึงค่าตัวแปรนั้นๆ

2.2.4) ส่วน condition else สุดท้าย หากไม่มีข้อมูลใดๆ ให้แสดง output ค่าเริ่มต้นเป็น 0

2.1.2) หลังจากนั้นเช็คลูป if ตามภาษีขั้นบันได โดยแสดง output เป็น %



รูป 3.23 โค้ดคำนวณ “สรุป” ส่วนแสดง “รายได้”

โค้ดคำอธิบายรูป 3.23 (ส่วนแสดงรายได้)

2.2.1) ในส่วนแรก ใช้คำสั่ง if เช็คตัวแปร arrayincomes ที่เก็บค่าใน Database (Model) Income ทั้งหมด ถ้าไม่เป็น nil ให้เข้าลูป

2.2.2) หลังจากนั้นเช็คลูป if โดยเช็ค ถ้าค่าเป็น nil ให้เข้าลูป if หรือ else ตามcondition นั้นๆ หลังจากนั้นแสดง output โดยดึงค่าจากตัวแปร sumin

2.2.3) ต่อมาเป็นส่วนแสดง รายจ่าย โดยคิดจาก sumin * 40% หาก output เกิน 60,000 บาทให้ตัวแปร expense = 60000 และแสดง outputโดยดึงค่าตัวแปรจาก expense และส่วนแสดง ลดหย่อนและเงินได้สุทธิ หลังจากคำนวณตามทฤษฎี ให้แสดง output โดยดึงค่าตัวแปรนั้นๆ

2.2.4) ส่วน condition else สุดท้าย หากไม่มีข้อมูลใดๆ ให้แสดง output ค่าเริ่มต้นเป็น 0

3) โค้ดคำนวณในส่วนของ “รายได้” ยกตัวอย่างบางส่วน มาดังนี้

```
let add = Income()

    if let check = amounts{ // ถ้า ค่าใน index Income ไม่เป็น nil
    var new : Double!
    let id = amount!.id
    let up = array[id-1]
    if checkNew == "salary"{
        new = check - amount!.salary
        new = new + incomeseg
        realm.beginWrite()
        up.salary = incomeseg
        up.all = new
        realm.add(up,update: true)
        try! realm.commitWrite()
        print("\(array)")
    }
    else if checkNew == "commission"{
        new = check - amount!.commission
        new = new + incomeseg
        realm.beginWrite()
        up.commission = incomeseg
        up.all = new
        realm.add(up,update: true)
        try! realm.commitWrite()
        print("\(array)")

    }
```

รูป 3.24 โค้ดคำนวณ “รายได้”

คำอธิบายโค้ดรูป 3.24

3.1) สร้างตัวแปรในการเก็บค่ามีชนิดเป็น Income() หรือ Model หลังจากนั้นทำการเทียบค่า โดยใช้คำสั่ง if let หากตัวแปรชื่อ “amounts” ไม่เป็น nil

3.2) สร้างตัวแปรเก็บค่าชนิด Double ชื่อ new และสร้างตัวแปรในการเรียก Object ที่มี Array ลำดับเดียวกับที่ต้องการแก้ไข ชื่อ up โดย id ให้อ้างอิงจาก index ช่องที่เราจะแก้ไข

3.3) โดยในหน้านี้ จะมีให้ผู้ใช้กรอกข้อมูลทั้งหมด 4 ตารางคือ เงินเดือน, ค่านายหน้า, ค่าเช่า, รายได้อื่นๆ เพราะฉะนั้นจะทำการเช็คว่า ได้รับ input จากหน้าไหนมาโดยยกตัวอย่าง จาก checkNew == “salaray” คือหน้ารายได้

3.4) ภายในลูปเป็นการคำนวณ โดยนำค่าเก่า คือ amount.salary บวกกับค่าใหม่คือ incomeseg หลังจากนั้นทำการ assign ค่าลงในตัวแปรที่มีชนิดเป็น Model ที่จะทำการ add หลังจากนั้นใช้คำสั่งในการ add

3) โค้ดคำนวณในส่วนของ “รายได้” ยกตัวอย่างบางส่วน มาดังนี้

```
let add = Income()

    if let check = amounts{ // [illegible]
    var new : Double!
    let id = amount!.id
    let up = array[id-1]
    if checkNew == "salary"{
        new = check - amount!.salary
        new = new + incomeseg
        realm.beginWrite()
        up.salary = incomeseg
        up.all = new
        realm.add(up,update: true)
        try! realm.commitWrite()
        print("\(array)")
    }
    } else if checkNew == "commission"{
        new = check - amount!.commission
        new = new + incomeseg
        realm.beginWrite()
        up.commission = incomeseg
        up.all = new
        realm.add(up,update: true)
        try! realm.commitWrite()
        print("\(array)")

    }
```

รูป 3.24 โค้ดคำนวณ “รายได้”

คำอธิบายโค้ดรูป 3.24

3.1) สร้างตัวแปรในการเก็บค่ามีชนิดเป็น Income() หรือ Model หลังจากนั้นทำการเทียบค่า โดยใช้คำสั่ง if let หากตัวแปรชื่อ “amounts” ไม่เป็น nil

3.2) สร้างตัวแปรเก็บค่าชนิด Double ชื่อ new และสร้างตัวแปรในการเรียก Object ที่มี Array ลำดับเดียวกับที่ต้องการแก้ไข ชื่อ up โดย id ให้อ้างอิงจาก index ช่องที่เราจะแก้ไข

3.3) โดยในหน้านี้ จะมีให้ผู้ใช้กรอกข้อมูลทั้งหมด 4 ตารางคือ เงินเดือน, ค่านายหน้า, ค่าเช่า, รายได้อื่นๆ เพราะฉะนั้นจะทำการเช็คว่า ได้รับ input จากหน้าไหนมาโดยยกตัวอย่าง จาก checkNew == “salaray” คือหน้ารายได้

3.4) ภายในลูปเป็นการคำนวณ โดยนำค่าเก่า คือ amount.salary บวกกับค่าใหม่คือ incomeseg หลังจากนั้นทำการ assign ค่าลงในตัวแปรที่มีชนิดเป็น Model ที่จะทำการ add หลังจากนั้นใช้คำสั่งในการ add

4) โค้ดคำนวณในส่วนของ “เป้าหมายการเงิน” ยกตัวอย่างบางส่วน มาดังนี้

```
else{ // ค่า index เงินทั้งหมด เป็น nil
    let financeNotValue = Finance()
    financeNotValue.id = financeNotValue.IncrementaID()
    financeNotValue.all = (financeseg.amount)
    financeNotValue.name = financeseg.name
    financeNotValue.amount = financeseg.amount
    financeNotValue.time = financeseg.time
    financeNotValue.interest = financeseg.interest
    // Add to database
    try! realm.write {
        realm.add(financeNotValue)
    }
    TagAll.text = financeNotValue.all.formatUsingAbbrevation()
    print("NOVALUE ")
}
```

รูป 3.25 โค้ดคำนวณ “เป้าหมายการเงิน” ในส่วนของเพิ่มเป้าหมาย

คำอธิบายโค้ดรูป 3.25 (ในส่วนของเพิ่มเป้าหมาย)

4.1.1) ภายในลูป else จะแสดงตอน add ข้อมูลในกรณีที่ไม่มี ค่าใน Model มาก่อน

4.1.2) สร้างตัวแปรในการเก็บค่ามีชนิดเป็น Finance() หรือ Model

4.1.3) ภายในลูปเป็นการคำนวณ โดยทำการบวกลำดับ id ใน database หลังจากนั้น ทำการ assign ค่าลงในตัวแปรที่มีชนิดเป็น Model ที่จะทำการ add หลังจากนั้น ใช้คำสั่งในการ add

4) โค้ดคำนวณในส่วนของ “เป้าหมายการเงิน” ยกตัวอย่างบางส่วน มาดังนี้

```
else{ // ค่า financeNotValue เป็น nil
    var financeNotValue = Finance()
    financeNotValue.id = financeNotValue.IncrementID()
    financeNotValue.all = (financeseg.amount)
    financeNotValue.name = financeseg.name
    financeNotValue.amount = financeseg.amount
    financeNotValue.time = financeseg.time
    financeNotValue.interest = financeseg.interest
    // Add to database
    try! realm.write {
        realm.add(financeNotValue)
    }
    TagAll.text = financeNotValue.all.formatUsingAbbrevation()
    print("nilvalue")
}
```

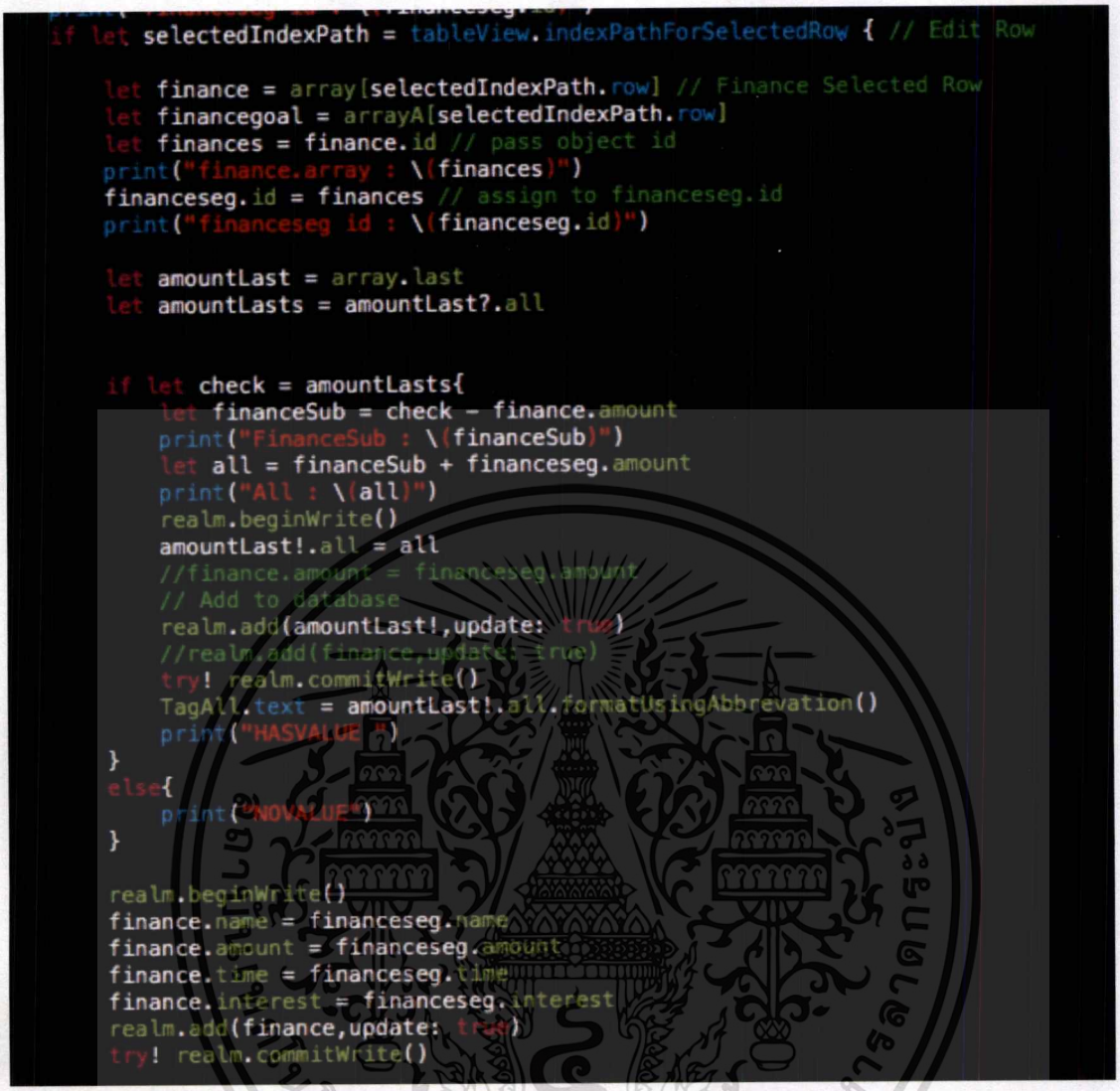
รูป 3.25 โค้ดคำนวณ “เป้าหมายการเงิน” ในส่วนของเพิ่มเป้าหมาย

คำอธิบายโค้ดรูป 3.25 (ในส่วนของเพิ่มเป้าหมาย)

4.1.1) ภายในลูป else จะแสดงตอน add ข้อมูลในกรณีที่ไม่มี ค่าใน Model มาก่อน

4.1.2) สร้างตัวแปรในการเก็บค่ามีชนิดเป็น Finance() หรือ Model

4.1.3) ภายในลูปเป็นการคำนวณ โดยทำการบวกลำดับ id ใน database หลังจากนั้น ทำการ assign ค่าลงในตัวแปรที่มีชนิดเป็น Model ที่จะทำการ add หลังจากนั้น ใช้คำสั่งในการ add



```
if let selectedIndexPath = tableView.indexPathForSelectedRow { // Edit Row

    let finance = array[selectedIndexPath.row] // Finance Selected Row
    let financegoal = arrayA[selectedIndexPath.row]
    let finances = finance.id // pass object id
    print("finance.array : \(finances)")
    financeseg.id = finances // assign to financeseg.id
    print("financeseg id : \(financeseg.id)")

    let amountLast = array.last
    let amountLasts = amountLast?.all


    if let check = amountLasts{
        let financeSub = check - finance.amount
        print("FinanceSub : \(financeSub)")
        let all = financeSub + financeseg.amount
        print("All : \(all)")
        realm.beginWrite()
        amountLast!.all = all
        //finance.amount = financeseg.amount
        // Add to database
        realm.add(amountLast!,update: true)
        //realm.add(finance,update: true)
        try! realm.commitWrite()
        TagAll.text = amountLast!.all.formatUsingAbbrevation()
        print("HASVALUE ")
    }
    else{
        print("NOVALUE")
    }

    realm.beginWrite()
    finance.name = financeseg.name
    finance.amount = financeseg.amount
    finance.time = financeseg.time
    finance.interest = financeseg.interest
    realm.add(finance,update: true)
    try! realm.commitWrite()
```

รูป 3.26 โค้ดคำนวณ “เป้าหมายการเงิน” ในส่วนของแก้ไขเป้าหมาย

คำอธิบายโค้ดรูป 3.26 (ในส่วนของแก้ไขเป้าหมาย)

4.2.1) ภายในลูปแรก บอกถึงถ้าผู้ใช้คลิกที่ Table นั้นๆให้ assign ค่าให้ selectedIndexPath

4.2.2) สร้างตัวแปรในการเก็บค่ามีชนิดเป็น Finance() หรือ Model ชื่อ finance โดยรับอันดับ row จากตัวแปร selectedIndexpath

4.2.3) หลังจากนั้น assign ค่า id ให้ตัวแปรชนิด Finance ที่จะทำการ update ค่า

4.2.4) เช็ค if let หากใน index finance ไม่เป็น nil

4.2.5) ภายในลูปทำการคำนวณ จากนั้นทำการ add ลง database

```
if let selectedIndexPath = tableView.indexPathForSelectedRow { // Edit Row

    let finance = array[selectedIndexPath.row] // Finance Selected Row
    let financegoal = arrayA[selectedIndexPath.row]
    let finances = finance.id // pass object id
    print("finance.array : \(finances)")
    financeseg.id = finances // assign to financeseg.id
    print("financeseg id : \(financeseg.id)")

    let amountLast = array.last
    let amountLasts = amountLast?.all


    if let check = amountLasts{
        let financeSub = check - finance.amount
        print("FinanceSub : \(financeSub)")
        let all = financeSub + financeseg.amount
        print("All : \(all)")
        realm.beginWrite()
        amountLast!.all = all
        //finance.amount = financeseg.amount
        // Add to database
        realm.add(amountLast!,update: true)
        //realm.add(finance,update: true)
        try! realm.commitWrite()
        TagAll.text = amountLast!.all.formatUsingAbbrevation()
        print("HASVALUE ")
    }
    else{
        print("NOVALUE")
    }

    realm.beginWrite()
    finance.name = financeseg.name
    finance.amount = financeseg.amount
    finance.time = financeseg.time
    finance.interest = financeseg.interest
    realm.add(finance,update: true)
    try! realm.commitWrite()
```

รูป 3.26 โค้ดคำนวณ “เป้าหมายการเงิน” ในส่วนของแก้ไขเป้าหมาย

คำอธิบายโค้ดรูป 3.26 (ในส่วนของแก้ไขเป้าหมาย)

4.2.1) ภายในลูปแรก บอกถึงถ้าผู้ใช้คลิกที่ Table นั้นๆให้ assign ค่าให้ selectedIndexPath

4.2.2) สร้างตัวแปรในการเก็บค่ามีชนิดเป็น Finance() หรือ Model ชื่อ finance โดยรับอันดับ row จากตัวแปร selectedIndexpath

4.2.3) หลังจากนั้น assign ค่า id ให้ตัวแปรชนิด Finance ที่จะทำการ update ค่า

4.2.4) เช็ค if let หากใน index finance ไม่เป็น nil

4.2.5) ภายในลูปทำการคำนวณ จากนั้นทำการ add ลง database

```
func handleDeletePlanet(alertAction: UIAlertAction!) -> Void {
    if let indexPath = deleteIndexPath {
        tableView.beginUpdates()

        //            planets.removeAtIndex(indexPath.row)

        let name = array.last
        let names = name?.all
        print("old fin all : \(names)")

        let goal = arrayA.last
        let goals = goal?.all
        print("old fin goal all : \(names)")

        let selectedFin = array[indexPath.row]
        let selectedFinGoal = arrayA[indexPath.row]
        let selectedFins = selectedFin.amount
        let selectedFinGoals = selectedFinGoal.pmt
        print("selected all : \(selectedFins)")
        print("selected fin all : \(selectedFinGoals)")

        let sum = names! - selectedFins //sum after delete row
        let sums = goals! - selectedFinGoals
        print("fin sum : \(sum)")
        print("fin goal sum : \(sums)")
        print("index \(indexPath.row)")
        print("count : \(array.count)")


        realm.beginWrite()
        realm.delete(array[indexPath.row])
        try! realm.commitWrite()
```

รูป 3.27 โค้ดคำนวณ "เป้าหมายการเงิน" ในส่วนของลบเป้าหมาย

คำอธิบายโค้ดรูป 3.27 (ในส่วนของลบเป้าหมาย)

4.3.1) ภายในลูปแรก บอกถึงถ้าผู้ใช้คลิกที่ Table นั้นๆให้ assign ค่า deleteIndexPath คือ path ที่เราต้องการจะลบ ให้ indexPath

4.3.2) หลังจากนั้น assign row ที่ต้องการจะทำการลบ ในตัวแปร selectedFin

4.3.3) หลังจากนั้น delete แถวนั้นๆ

```
handleDeletePlanet(alertAction: UIAlertAction!) -> Void {
    indexPath = deleteIndexPath {
    tableView.beginUpdates()

    //            planets.removeAtIndex(indexPath.row)

    name = array.last
    names = name?.all
    print("จำนวนเงินทั้งหมด : \(names)")

    goal = arrayA.last
    goals = goal?.all
    print("เป้าหมายการเงินทั้งหมด : \(names)")

    selectedFin = array[indexPath.row]
    selectedFinGoal = arrayA[indexPath.row]
    selectedFins = selectedFin.amount
    selectedFinGoals = selectedFinGoal.amt
    print("จำนวนเงินที่ลบ : \(selectedFins)")
    print("เป้าหมายที่ลบ : \(selectedFinGoals)")

    sum = names! - selectedFins //ยอดเงินหลังจากลบ
    sums = goals! - selectedFinGoals
    print("ยอดเงินใหม่ : \(sum)")
    print("เป้าหมายใหม่ : \(sums)")
    print("ลำดับที่ \(indexPath.row)")
    print("จำนวน : \(array.count)")


    realm.beginWrite()
    realm.delete(array[indexPath.row])
    try! realm.commitWrite()
```

รูป 3.27 โค้ดคำนวณ "เป้าหมายการเงิน" ในส่วนของลบเป้าหมาย

คำอธิบายโค้ดรูป 3.27 (ในส่วนของลบเป้าหมาย)

4.3.1) ภายในลูปแรก บอกถึงถ้าผู้ใช้คลิกที่ Table นั้นๆให้ assign ค่า deleteIndexPath คือ path ที่เราต้องการจะลบ ให้ indexPath

4.3.2) หลังจากนั้น assign row ที่ต้องการจะทำการลบ ในตัวแปร selectedFin

4.3.3) หลังจากนั้น delete แถวนั้นๆ

5) โค้ดคำนวณในส่วนของ “ภาษี” ยกตัวอย่างบางส่วน มาดังนี้

```
// check ค่า all ว่าเท่ากับเท่าไร
let add = Tax()

if let check = amounts{ // ถ้า ค่าใน index Income ไม่เป็น nil
    var new : Double!
    let id = amount!.id
    let up = arrayTax[id-1]
    if checkNew == "ltf"{
        if let a = maxlm{
            if incomeseg > a{
                incomeseg = a
            }
        }
        new = check - amount!.LTF
        new = new + incomeseg
        realm.beginWrite()
        up.LTF = incomeseg
        up.all = new
        realm.add(up,update: true)
        try! realm.commitWrite()
        print("\(arrayTax)")
    }
    else if checkNew == "rmf"{
        if let a = maxlm{
            if incomeseg > a{
                incomeseg = a
            }
        }
        new = check - amount!.RMF
        new = new + incomeseg
        realm.beginWrite()
        up.RMF = incomeseg
        up.all = new
        realm.add(up,update: true)
        try! realm.commitWrite()
        print("\(arrayTax)")
    }
```

รูป 3.28 โค้ดคำนวณ “ภาษี”

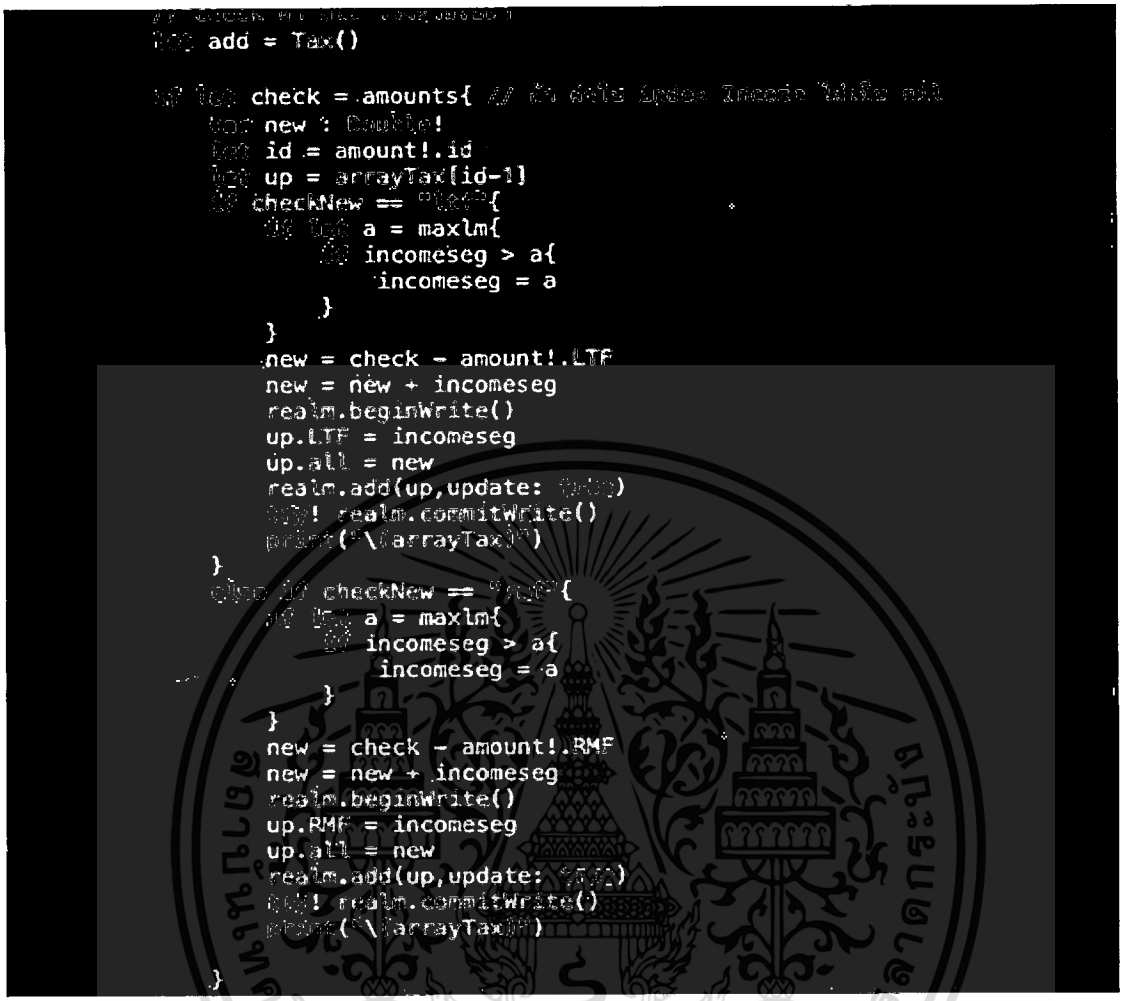
คำอธิบายโค้ดรูป 3.28

5.1) สร้างตัวแปรในการเก็บค่ามีชนิดเป็น Tax() หรือ Model หลังจากนั้นทำการเทียบค่า โดยใช้คำสั่ง if let หากตัวแปรชื่อ “amounts” ไม่เป็น nil

5.2) สร้างตัวแปรเก็บค่าชนิด Double ชื่อ new และสร้างตัวแปรในการเรียก Object ที่มี Array ลำดับเดียวกับที่ต้องการแก้ไข ชื่อ up โดย id ให้อ้างอิงจาก index ช่องที่เราจะแก้ไข

5.3) โดยในหน้านี้ จะมีให้ผู้ใช้กรอกข้อมูลทั้งหมด 4 ตารางคือ LTF, RMF, เบี้ยประกันชีวิตทั่วไป, เบี้ยประกันชีวิตแบบบำนาญ เพราะฉะนั้นจะทำการเช็คว่าได้รับ input จากหน้าไหนมา โดยยกตัวอย่าง จาก checkNew == “ltf” คือหน้า LTF

5) โค้ดคำนวณในส่วนของ “ภาษี” ยกตัวอย่างบางส่วน มาดังนี้



```
add = Tax()

check = amounts{
new : Double!
id = amount!.id
up = arrayTax[id-1]
checkNew == {
    a = maxIm{
        incomeseg > a{
            incomeseg = a
        }
    }
    new = check - amount!.LTF
    new = new + incomeseg
    realm.beginWrite()
    up.LTF = incomeseg
    up.all = new
    realm.add(up,update:
    ! realm.commitWrite()
    print("\(arrayTax)")
}
    checkNew == {
        a = maxIm{
            incomeseg > a{
                incomeseg = a
            }
        }
        new = check - amount!.RMF
        new = new + incomeseg
        realm.beginWrite()
        up.RMF = incomeseg
        up.all = new
        realm.add(up,update:
        ! realm.commitWrite()
        print("\(arrayTax)")
}
```

รูป 3.28 โค้ดคำนวณ “ภาษี”

คำอธิบายโค้ดรูป 3.28

5.1) สร้างตัวแปรในการเก็บค่ามีชนิดเป็น Tax() หรือ Model หลังจากนั้นทำการเทียบค่า โดยใช้คำสั่ง if let หากตัวแปรชื่อ “amounts” ไม่เป็น nil

5.2) สร้างตัวแปรเก็บค่าชนิด Double ชื่อ new และสร้างตัวแปรในการเรียก Object ที่มี Array ลำดับเดียวกับที่ต้องการแก้ไข ชื่อ up โดย id ให้อ้างอิงจาก index ช่องที่เราจะแก้ไข

5.3) โดยในหน้านี้ จะมีให้ผู้ใช้กรอกข้อมูลทั้งหมด 4 ตารางคือ LTF, RMF, เบี้ยประกันชีวิตทั่วไป, เบี้ยประกันชีวิตแบบบำนาญ เพราะฉะนั้นจะทำการเช็คว่าได้รับ input จากหน้าไหนมา โดยยกตัวอย่าง จาก checkNew == “ltf” คือหน้า LTF

5.4) ภายในลูปเป็นการคำนวณ assign ค่าลงในตัวแปรที่มีชนิดเป็น Model ที่จะทำการ add หลังจากนั้นใช้คำสั่งในการ add

3.9.3 โค้ดส่วน Model ทั้งหมด (Database)

1) โค้ดส่วน Tax Model

```
class Tax : Object {

    dynamic var id : Int = 0
    dynamic var LTF : Double = 0.0
    dynamic var RMF : Double = 0.0
    dynamic var ins : Double = 0.0
    dynamic var ins1 : Double = 0.0
    dynamic var all : Double = 0.0
    dynamic var willtax : Double = 0.0
    dynamic var maxltfrmf : Double = 0.0
    dynamic var maxins : Double = 0.0
    dynamic var maxins1 : Double = 0.0

    override static func primaryKey() -> String?
    {
        return "id"
    }

    func IncrementaID() -> Int{
        let realm = try! Realm()
        let RetNext: NSArray = Array(realm.objects(Tax).sorted("id"))
        let last = RetNext.lastObject
        if RetNext.count > 0 {
            let valor = last?.valueForKey("id") as? Int
            return valor! + 1
        } else {
            return 1
        }
    }
```

รูป 3.29 Tax Model

คำอธิบายโค้ดรูป 3.29 (Tax Database)

1.1) อันดับแรก ต้องทำการ import Realm บน class โดย Realm Database จะมีการประกาศตัวแปรเป็นชนิด Object

1.2) วิธีประกาศตัวแปร ให้ประกาศ ตัวแปร dynamic var ตามด้วยชื่อตัวแปร : ชนิดตัวแปร

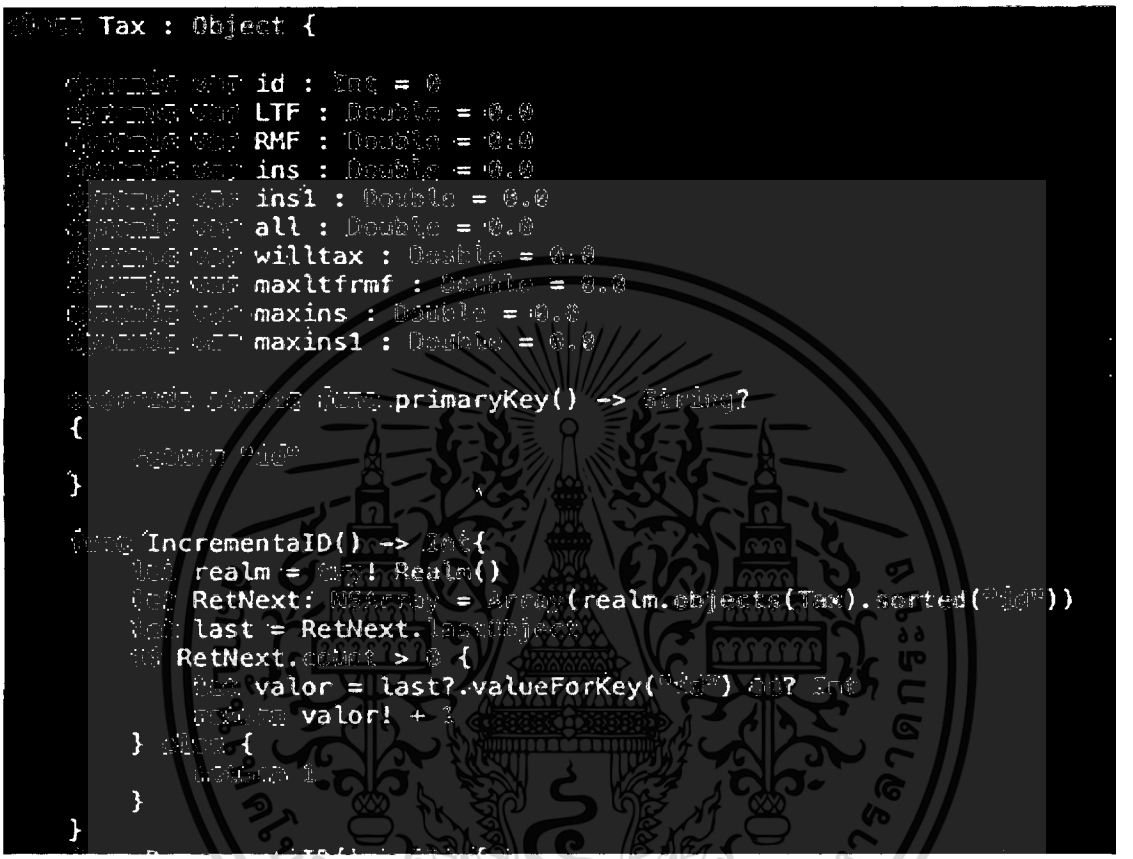
1.3) ฟังก์ชั่น primaryKey() คือการกำหนดคีย์ให้เป็น Primary Key อะไรให้ return ตัวแปรนั้นๆ

1.4) ฟังก์ชั่น IncrementID คือการ Increment ID บน Database ไว้ใช้สำหรับการ add ค่าลง Database นี้ วิธีการประกาศคือ tax.IncrementID

5.4) ภายในลูปเป็นการคำนวณ assign ค่าลงในตัวแปรที่มีชนิดเป็น Model ที่จะทำการ add หลังจากนั้นใช้คำสั่งในการ add

3.9.3 โค้ดส่วน Model ทั้งหมด (Database)

1) โค้ดส่วน Tax Model



รูป 3.29 Tax Model

คำอธิบายโค้ดรูป 3.29 (Tax Database)

1.1) อันดับแรก ต้องทำการ import Realm บน class โดย Realm Database จะมีการประกาศตัวแปรเป็นชนิด Object

1.2) วิธีประกาศตัวแปร ให้ประกาศ ตัวแปร dynamic var ตามด้วยชื่อตัวแปร : ชนิดตัวแปร

1.3) ฟังก์ชั่น primaryKey() คือการกำหนดคีย์ให้เป็น Primary Key อะไรให้ return ตัวแปรนั้นๆ

1.4) ฟังก์ชั่น IncrementID คือการ Increment ID บน Database ไว้ใช้สำหรับการ add ค่าลง Database นี้ วิธีการประกาศคือ tax.IncrementID

2) โค้ดส่วน Income Model

```
class Income : Object {

    dynamic var id : Int = 0
    dynamic var salary : Double = 0.0
    dynamic var commission : Double = 0.0
    dynamic var rent : Double = 0.0
    dynamic var other : Double = 0.0
    dynamic var all : Double = 0.0

    override static func primaryKey() -> String?
    {
        return "id"
    }


    func IncrementaID() -> Int{
        let realm = try! Realm()
        let RetNext: NSArray = Array(realm.objects(Income).sorted("id"))
        let last = RetNext.lastObject
        if RetNext.count > 0 {
            let valor = last?.valueForKey("id") as? Int
            return valor! + 1
        } else {
            return 1
        }
    }
```

รูป 3.30 Income Model

คำอธิบายโค้ดรูป 3.30 (Income Database)

2.1) อันดับแรก ต้องทำการ import Realm บน class โดย Realm Database จะมีการประกาศตัวแปรเป็นชนิด Object

2.2) วิธีประกาศตัวแปร ให้ประกาศ ตัวแปร dynamic var ตามด้วยชื่อตัวแปร : ชนิดตัวแปร

2.3) ฟังก์ชั่น primaryKey() คือการกำหนดคีย์ให้เป็น Primary Key อะไรให้ return ตัวแปรนั้นๆ

2.4) ฟังก์ชั่น IncrementID คือการ Increment ID บน Database ไว้ใช้สำหรับการ add ค่าลง Database นี้ วิธีการประกาศคือ tax.IncrementID

2) โค้ดส่วน Income Model

```
class Income : Object {

    @objc dynamic var id : Int = 0
    @objc dynamic var salary : Double = 0.0
    @objc dynamic var commission : Double = 0.0
    @objc dynamic var rent : Double = 0.0
    @objc dynamic var other : Double = 0.0
    @objc dynamic var all : Double = 0.0

    override static func primaryKey() -> String?
    {
        return "id"
    }


    static func IncrementaID() -> Int{
        let realm = try! Realm()
        let RetNext: Results<Income> = Array(realm.objects(Income).sorted(byKeyPath: "id"))
        let last = RetNext.last
        if RetNext.count > 0 {
            var valor = last?.value(forKey: "id") as? Int
            return valor! + 1
        } else {
            return 1
        }
    }
```

รูป 3.30 Income Model

คำอธิบายโค้ดรูป 3.30 (Income Database)

2.1) อันดับแรก ต้องทำการ import Realm บน class โดย Realm Database จะมีการประกาศตัวแปรเป็นชนิด Object

2.2) วิธีประกาศตัวแปร ให้ประกาศ ตัวแปร dynamic var ตามด้วยชื่อตัวแปร : ชนิดตัวแปร

2.3) ฟังก์ชั่น primaryKey() คือการกำหนดคีย์ให้เป็น Primary Key อะไรให้ return ตัวแปรนั้นๆ

2.4) ฟังก์ชั่น IncrementID คือการ Increment ID บน Database ไว้ใช้สำหรับการ add ค่าลง Database นี้ วิธีการประกาศคือ tax.IncrementID

3) โค้ดส่วน Finance Model

```
class Finance : Object {

    dynamic var id : Int = 0
    dynamic var name : String = ""
    dynamic var amount : Double = 0.0
    dynamic var time : Int = 1
    dynamic var interest : Double = 0.0
    dynamic var all : Double = 0.0

    override static func primaryKey() -> String?
    {
        return "id"
    }

    func IncrementaID() -> Int{
        let realm = try! Realm()
        let RetNext: NSArray = Array(realm.objects(Finance).sorted("id"))
        let last = RetNext.lastObject
        if RetNext.count > 0 {
            let valor = last?.valueForKey("id") as? Int
            return valor! + 1
        } else {
            return 1
        }
    }
```

รูป 3.31 Finance Model

คำอธิบายโค้ดรูป 3.31 (Finance Database)

3.1) อันดับแรก ต้องทำการ import Realm บน class โดย Realm Database จะมีการประกาศตัวแปรเป็นชนิด Object

3.2) วิธีประกาศตัวแปร ให้ประกาศ ตัวแปร dynamic var ตามด้วยชื่อตัวแปร : ชนิดตัวแปร

3.3) ฟังก์ชั่น primaryKey() คือการกำหนดคีย์ให้เป็น Primary Key อะไรให้ return ตัวแปรนั้นๆ

3.4) ฟังก์ชั่น IncrementID คือการ Increment ID บน Database ไว้ใช้สำหรับการ add ค่าลง Database นี้ วิธีการประกาศคือ tax.IncrementID

3) โค้ดส่วน Finance Model

รูป 3.31 Finance Model

คำอธิบายโค้ดรูป 3.31 (Finance Database)

3.1) อันดับแรก ต้องทำการ import Realm บน class โดย Realm Database จะมีการประกาศตัวแปรเป็นชนิด Object

3.2) วิธีประกาศตัวแปร ให้ประกาศ ตัวแปร dynamic var ตามด้วยชื่อตัวแปร : ชนิดตัวแปร

3.3) ฟังก์ชั่น primaryKey() คือการกำหนดคีย์ให้เป็น Primary Key อะไรให้ return ตัวแปรนั้นๆ

3.4) ฟังก์ชั่น IncrementID คือการ Increment ID บน Database ไว้ใช้สำหรับการ add ค่าลง Database นี้ วิธีการประกาศคือ tax.IncrementID

4) โค้ดส่วน Finance Goal Model

```
class FinanceGoal : Object {

    dynamic var id : Int = 0
    dynamic var income : Double = 0.0
    dynamic var gamount : Double = 0.0
    dynamic var interest : Double = 0.0
    dynamic var time : Int = 0
    dynamic var pmt : Double = 0.0
    dynamic var all : Double = 0.0

    override static func primaryKey() -> String?
    {
        return "id"
}

    func IncrementaID() -> Int{
        let realm = try! Realm()
        let RetNext: NSArray = Array(realm.objects(FinanceGoal).sorted("id"))
        let last = RetNext.lastObject
        if RetNext.count > 0 {
            let valor = last?.valueForKey("id") as? Int
            return valor! + 1
        } else {
            return 1
        }
    }
```

รูป 3.32 Finance Goal Model

คำอธิบายโค้ดรูป 3.32 (Finance Goal Database)

4.1) อันดับแรก ต้องทำการ import Realm บน class โดย Realm Database จะมีการประกาศตัวแปรเป็นชนิด Object

4.2) วิธีประกาศตัวแปร ให้ประกาศ ตัวแปร dynamic var ตามด้วยชื่อตัวแปร : ชนิดตัวแปร

4.3) ฟังก์ชั่น primaryKey() คือการกำหนดคีย์ให้เป็น Primary Key อะไรให้ return ตัวแปรนั้นๆ

4.4) ฟังก์ชั่น IncrementID คือการ Increment ID บน Database ไว้ใช้สำหรับการ add ค่าลง Database นี้ วิธีการประกาศคือ tax.IncrementID

4) โค้ดส่วน Finance Goal Model

```
class FinanceGoal : Object {

    @objc dynamic var id : Int = 0
    @objc dynamic var income : Double = 0.0
    @objc dynamic var gamount : Double = 0.0
    @objc dynamic var interest : Double = 0.0
    @objc dynamic var time : Int = 0
    @objc dynamic var pmt : Double = 0.0
    @objc dynamic var all : Double = 0.0

    override static func primaryKey() -> String?
    {
        return "id"
    }

    func IncrementaID() -> Int{
        let realm = try! Realm()
        let RetNext: Array = Array(realm.objects(FinanceGoal).sorted("id"))
        let last = RetNext.last
        if RetNext.count > 0 {
            let valor = last?.value(forKey: "id") as? Int
            return valor! + 1
        } else {
            return 1
        }
    }
```

รูป 3.32 Finance Goal Model

คำอธิบายโค้ดรูป 3.32 (Finance Goal Database)

4.1) อันดับแรก ต้องทำการ import Realm บน class โดย Realm Database จะมีการประกาศตัวแปรเป็นชนิด Object

4.2) วิธีประกาศตัวแปร ให้ประกาศ ตัวแปร dynamic var ตามด้วยชื่อตัวแปร : ชนิดตัวแปร

4.3) ฟังก์ชั่น primaryKey() คือการกำหนดคีย์ให้เป็น Primary Key อะไรให้ return ตัวแปรนั้นๆ

4.4) ฟังก์ชั่น IncrementID คือการ Increment ID บน Database ไว้ใช้สำหรับการ add ค่าลง Database นี้ วิธีการประกาศคือ tax.IncrementID

5) โค้ดส่วน Finance Health Model

```
class FinanceHealth : Object {

    dynamic var id : Int = 0
    dynamic var sinsuball: Double = 0.0
    dynamic var sinsublongtun : Double = 0.0
    dynamic var sinsubsapab : Double = 0.0
    dynamic var sinsubprivate : Double = 0.0

    dynamic var nheesinall : Double = 0
    dynamic var nheesinlow : Double = 0.0
    dynamic var nheesinhigh : Double = 0.0

    dynamic var all : Double = 0.0
    override static func primaryKey() -> String?
    {
        return "id"
    }

    func IncrementaID() -> Int{
        let realm = try! Realm()
        let RetNext: NSArray = Array(realm.objects(FinanceHealth).sorted("id"))
        let last = RetNext.lastObject
        if RetNext.count > 0 {
            let valor = last?.valueForKey("id") as? Int
            return valor! + 1
        } else {
            return 1
        }
    }
```

รูป 3.33 Finance Health Model

คำอธิบายโค้ดรูป 3.33 (Finance Health Database)

5.1) อันดับแรก ต้องทำการ import Realm บน class โดย Realm Database จะมีการประกาศตัวแปรเป็นชนิด Object

5.2) วิธีประกาศตัวแปร ให้ประกาศ ตัวแปร dynamic var ตามด้วยชื่อตัวแปร : ชนิดตัวแปร

5.3) ฟังก์ชั่น primaryKey() คือการกำหนดคีย์ให้เป็น Primary Key อะไรให้ return ตัวแปรนั้นๆ

5.4) ฟังก์ชั่น IncrementID คือการ Increment ID บน Database ไว้ใช้สำหรับการ add ค่าลง Database นี้ วิธีการประกาศคือ tax.IncrementID

5) โค้ดส่วน Finance Health Model

```
FinanceHealth : Object {

    id : Int = 0
    sinsuball: Double = 0.0
    sinsublongtun : Double = 0.0
    sinsubsapab : Double = 0.0
    sinsubprivate : Double = 0.0

    nheesinall : Double = 0
    nheesinlow : Double = 0.0
    nheesinhigh : Double = 0.0

    all : Double = 0.0
    override fun primaryKey() -> String?
    {
    }

    IncrementaID() -> Int{
        realm = try! Realm()
        RetNext: Results = Array(realm.objects(FinanceHealth).sorted("id"))
        last = RetNext.last
        if RetNext.count > 0 {
            valor = last?.valueForKey("id") as? Int
            valor! + 1
        } else {
            1
        }
    }
```

รูป 3.33 Finance Health Model

คำอธิบายโค้ดรูป 3.33 (Finance Health Database)

5.1) อันดับแรก ต้องทำการ import Realm บน class โดย Realm Database จะมีการประกาศตัวแปรเป็นชนิด Object

5.2) วิธีประกาศตัวแปร ให้ประกาศ ตัวแปร dynamic var ตามด้วยชื่อตัวแปร : ชนิดตัวแปร

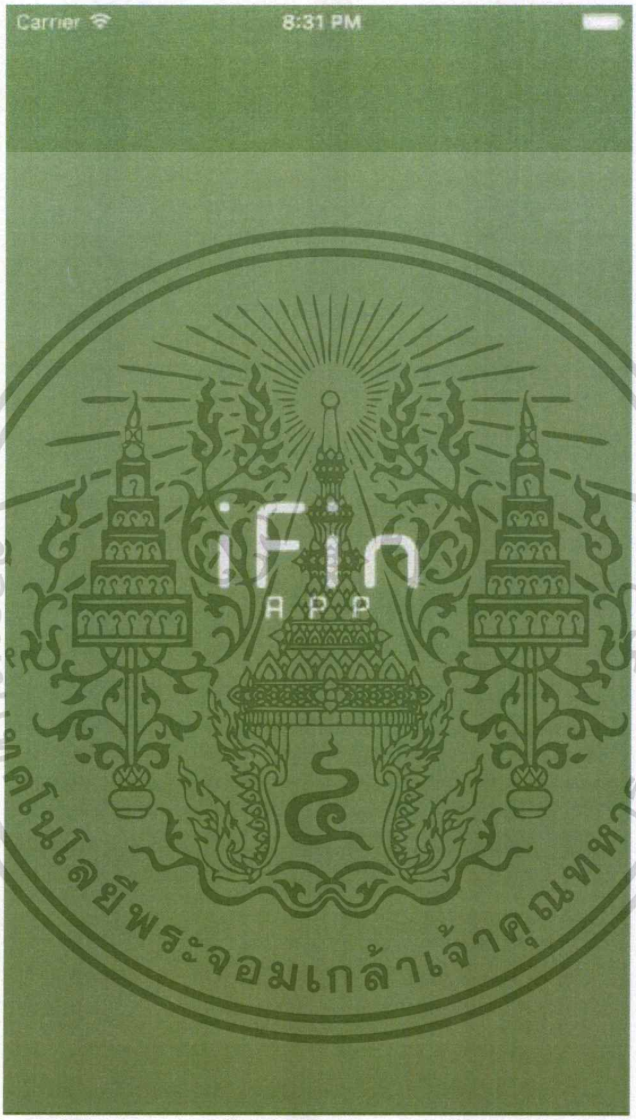
5.3) ฟังก์ชั่น primaryKey() คือการกำหนดคีย์ให้เป็น Primary Key อะไรให้ return ตัวแปรนั้นๆ

5.4) ฟังก์ชั่น IncrementID คือการ Increment ID บน Database ไว้ใช้สำหรับการ add ค่าลง Database นี้ วิธีการประกาศคือ tax.IncrementID

บทที่ 4

ผลการทดลอง

หน้าจอแรกเริ่มของแอพพลิเคชั่นการเงินส่วนบุคคล

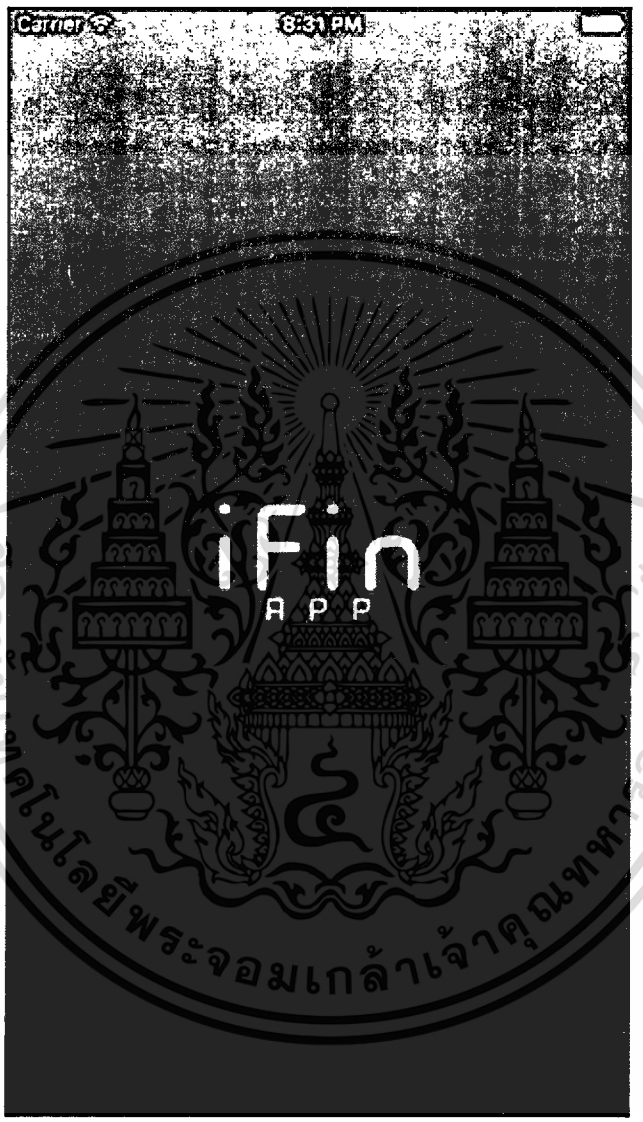


รูป 4.1 หน้าจอตอนเข้าสู่โปรแกรม

บทที่ 4

ผลการทดลอง

หน้าจอแรกเริ่มของแอพพลิเคชั่นการเงินส่วนบุคคล



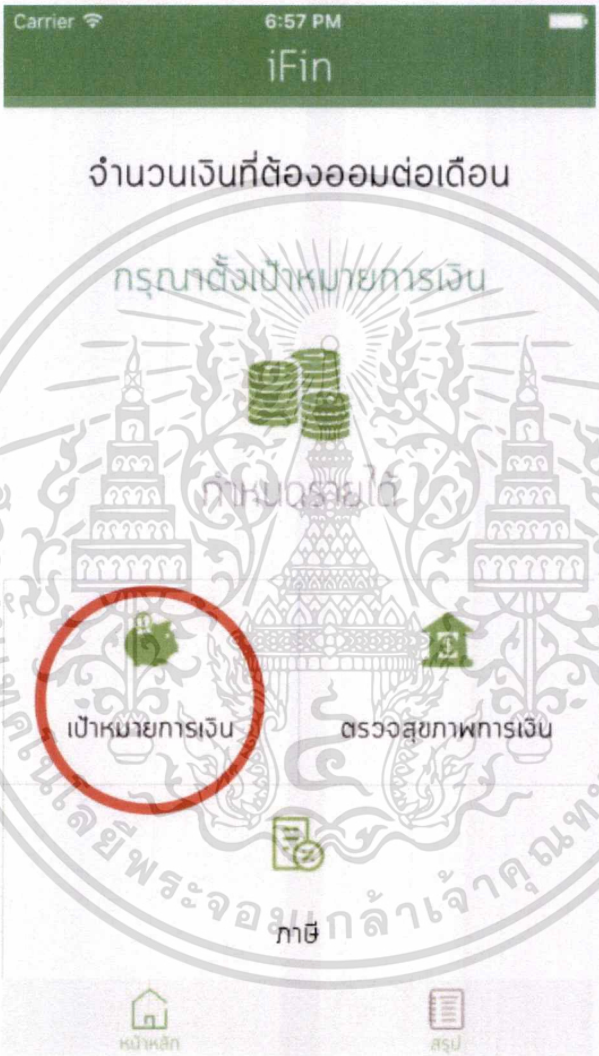
รูป 4.1 หน้าจอตอนเข้าสู่โปรแกรม

4.1 ผลการทดลองเป้าหมายทางการเงิน

จากการทดสอบฟังก์ชั่นเป้าหมายทางการเงิน เมื่อมีการใส่ค่าตัวแปรในเป้าหมายการเงิน คือ มูลค่าเป้าหมาย ระยะเวลาเก็บออม และดอกเบี้ย จะได้ค่าออมเงินในแต่ละเดือน

ส่วนการใช้งานฟังก์ชั่นเป้าหมายทางการเงินมีขั้นตอนดังนี้

1) ให้กดเข้าไปที่เป้าหมายทางการเงิน



รูป 4.2 หน้าจอหลักแอพพลิเคชั่น

4.1 ผลการทดลองเป้าหมายทางการเงิน

จากการทดสอบฟังก์ชั่นเป้าหมายทางการเงิน เมื่อมีการใส่ค่าตัวแปรในเป้าหมายการเงิน คือ มูลค่าเป้าหมาย ระยะเวลาเก็บออม และดอกเบี้ย จะได้ค่าออมเงินในแต่ละเดือน

ส่วนการใช้งานฟังก์ชั่นเป้าหมายทางการเงินมีขั้นตอนดังนี้

1) ให้กดเข้าไปที่เป้าหมายทางการเงิน

รูป 4.2 หน้าจอหลักแอพพลิเคชั่น

2) หน้าจอเป้าหมายทางการเงิน เมื่อต้องการใส่เป้าหมายทางการเงินให้กดเครื่องหมาย เพื่อเข้าสู่หน้าจอใส่ข้อมูลตั้งเป้าหมายทางการเงิน

รูป 4.3 หน้าจอเป้าหมายทางการเงิน

2) หน้าจอเป้าหมายทางการเงิน เมื่อต้องการใส่เป้าหมายทางการเงินให้กดเครื่องหมาย เพื่อเข้าสู่หน้าจอใส่ข้อมูลตั้งเป้าหมายทางการเงิน

รูป 4.3 หน้าจอเป้าหมายทางการเงิน

3) เป้าหมาย - ให้ใส่สิ่งที่เราจะเก็บออมเพื่อ

จำนวนเงิน - มูลค่าของเป้าหมายที่เราจะเก็บเงินเพื่อ

ระยะเวลา - จำนวนระยะเวลาที่ใช้เก็บออม

ดอกเบี้ย - จำนวนผลตอบแทนในสิ่งที่ทรัพย์สินที่ลงทุน

รูป 4.4 ใส่ข้อมูลตัวเป้าหมายของผู้ใช้

3) เป้าหมาย - ให้ใส่สิ่งที่เราจะเก็บออมเพื่อ

จำนวนเงิน - มูลค่าของเป้าหมายที่เราจะเก็บเงินเพื่อ

ระยะเวลา - จำนวนระยะเวลาที่ใช้เก็บออม

ดอกเบี้ย - จำนวนผลตอบแทนในสิ่งที่ทรัพย์สินที่ลงทุน

รูป 4.4 ใส่ข้อมูลตัวเป้าหมายของผู้ใช้

4) ยกตัวอย่าง เช่น เงินดาวน์บ้าน

เป้าหมาย = เงินดาวน์บ้าน

จำนวนเงิน = 100,000 บาท

ระยะเวลา = 5 ปี

ดอกเบี้ย = 3%

เมื่อเสร็จแล้วให้กด “Done”

รูป 4.5 ตัวอย่างการใส่ข้อมูลเป้าหมายการเงิน

4) ยกตัวอย่าง เช่น เงินดาวน์บ้าน

เป้าหมาย = เงินดาวน์บ้าน

จำนวนเงิน = 100,000 บาท

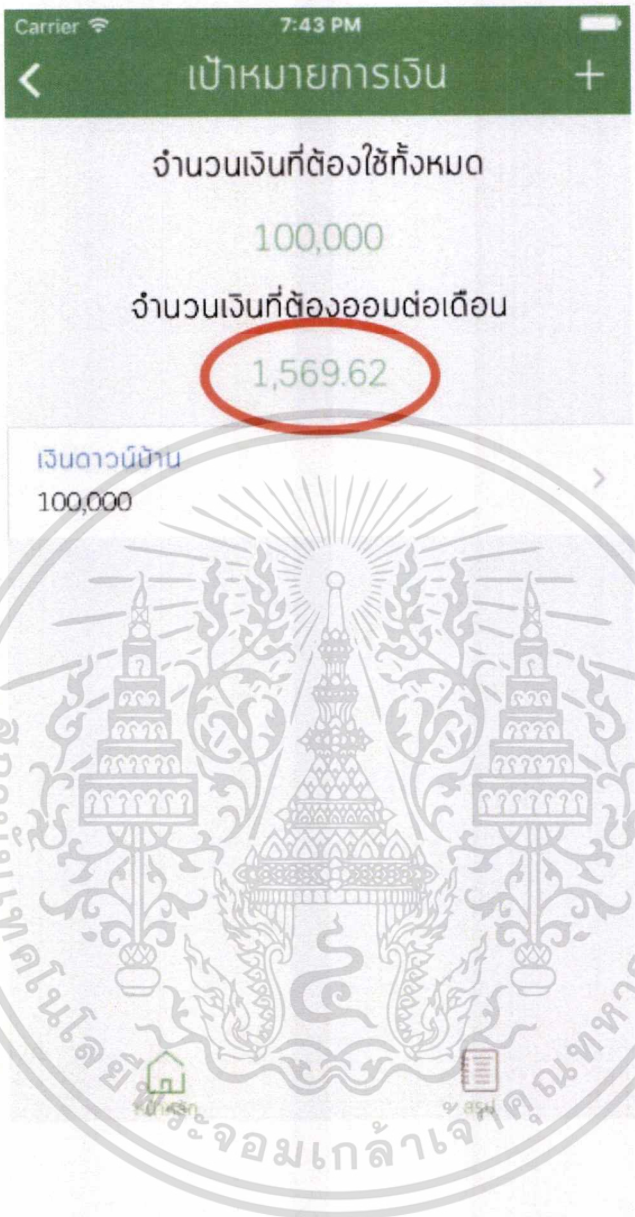
ระยะเวลา = 5 ปี

ดอกเบี้ย = 3%

เมื่อเสร็จแล้วให้กด “Done”

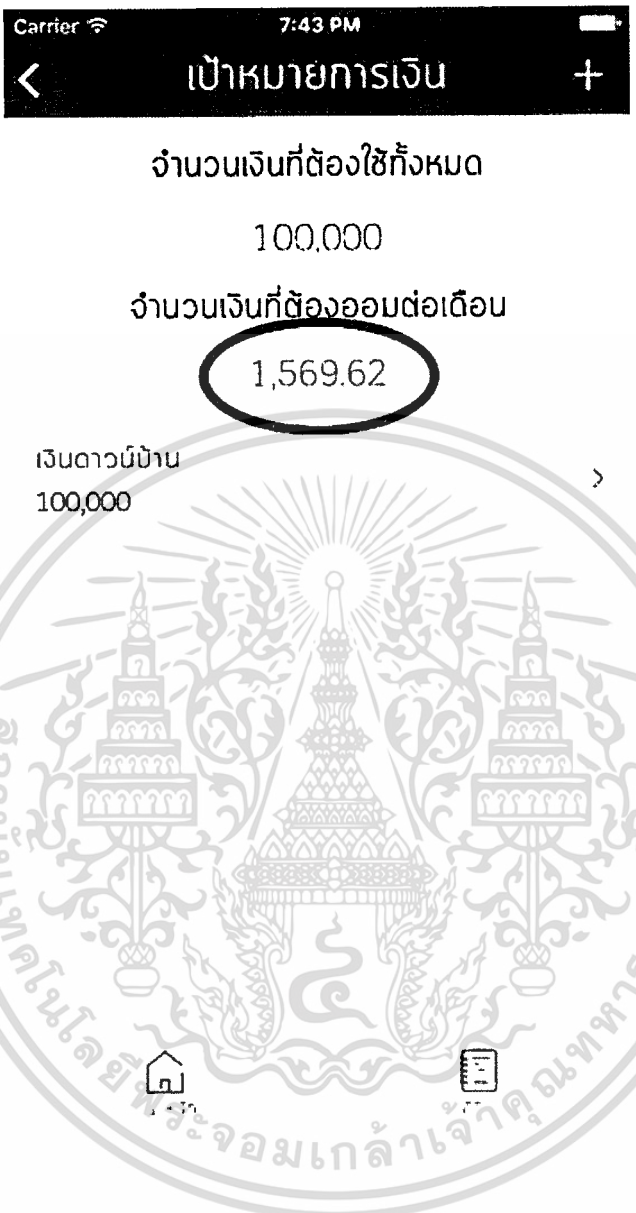
รูป 4.5 ตัวอย่างการใส่ข้อมูลเป้าหมายการเงิน

5) เมื่อใส่ค่าเป้าหมายทางการเงินเรียบร้อย จะได้เงินที่ต้องออมต่อเดือนออกมา



รูป 4.6 เป้าหมายการออมต่อเดือน

5) เมื่อใส่ค่าเป้าหมายทางการเงินเรียบร้อย จะได้เงินที่ต้องออมต่อเดือนออกมา

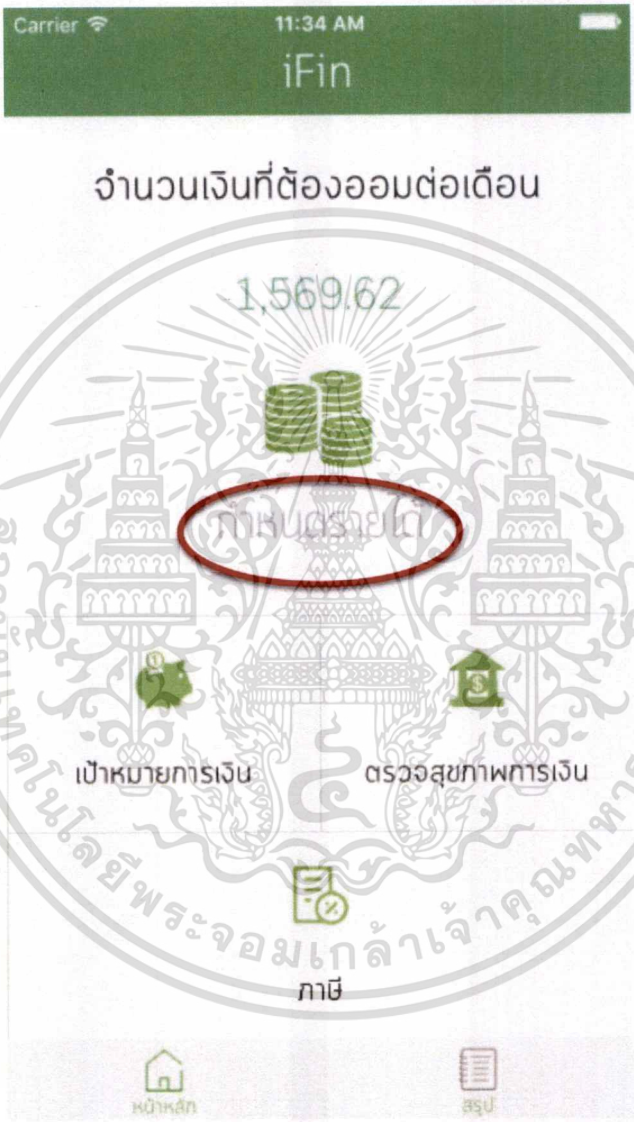


รูป 4.6 เป้าหมายการออมต่อเดือน

4.2 ภาษี

เมื่อใส่จำนวนรายได้ของเราก็จะคำนวณจำนวนที่จะเสียภาษีและค่าลดหย่อน LTF,RMF และประกันชีวิตแบบบำนาญได้

1) ฟังก์ชั่นภาษีต้องเริ่มจากการกำหนดรายได้ก่อนโดยกดคำว่า “กำหนดรายได้”



รูป 4.7 การกำหนดรายได้

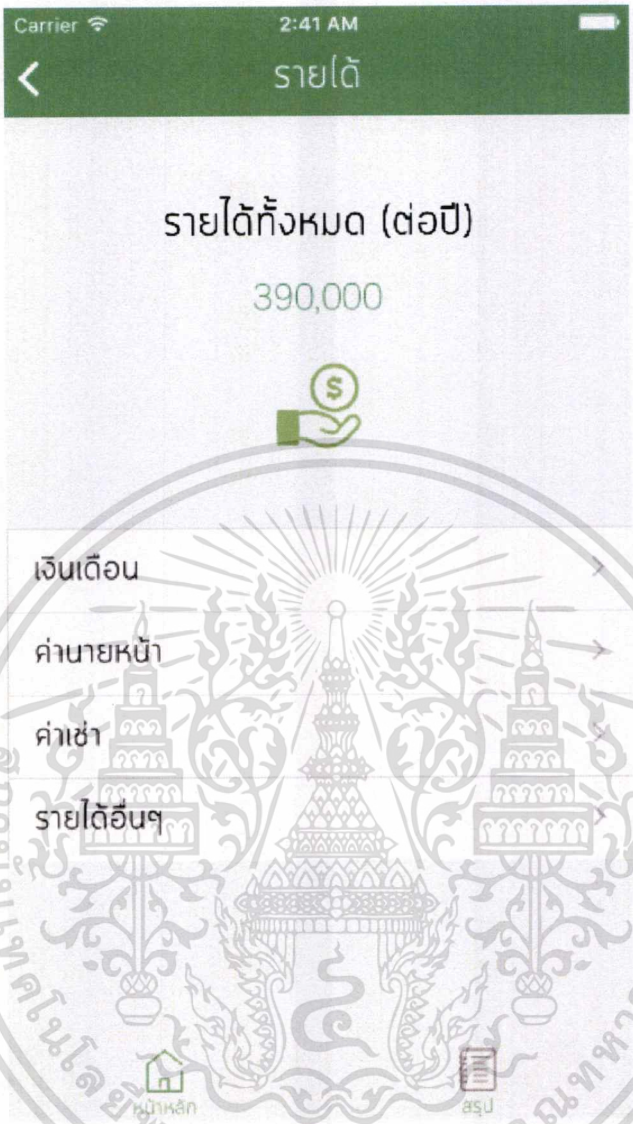
4.2 ภาษี

เมื่อใส่จำนวนรายได้ของเราก็จะคำนวณจำนวนที่จะเสียภาษีและค่าลดหย่อน LTF,RMF และประกันชีวิตแบบบำนาญได้

1) ฟังก์ชั่นภาษีต้องเริ่มจากการกำหนดรายได้ก่อนโดยกดคำว่า “กำหนดรายได้”

รูป 4.7 การกำหนดรายได้

2) กำหนดรายได้ของผู้ใช้โดยจะมี เงินเดือน, ค่านายหน้า, ค่าเช่า และรายได้อื่นๆ

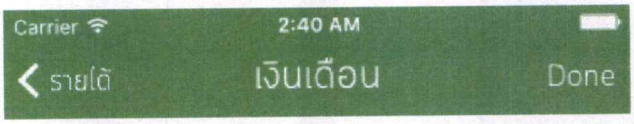


รูป 4.8 การกำหนดรายได้

2) กำหนดรายได้ของผู้ใช้โดยจะมี เงินเดือน, ค่านายหน้า, ค่าเช่า และรายได้อื่นๆ

รูป 4.8 การกำหนดรายได้

3) ใส่จำนวนเงินเดือนเป็นรายเดือนเพื่อคำนวนภาษี



เงินเดือน

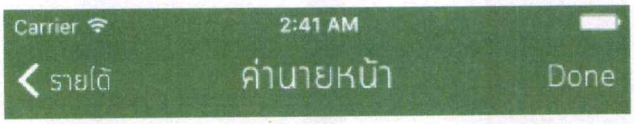
30000

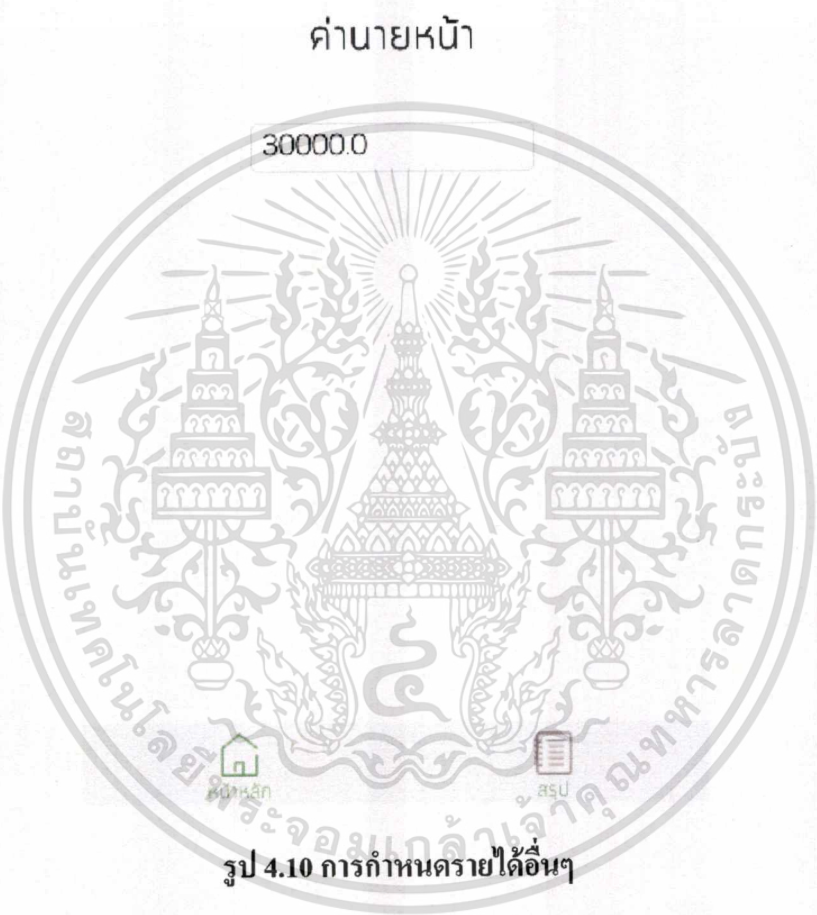
รูป 4.9 กำหนดเงินเดือน

3) ใส่จำนวนเงินเดือนเป็นรายเดือนเพื่อคำนวนภาษี

รูป 4.9 กำหนดเงินเดือน

4) ส่วนรายได้อื่นๆให้ใส่เป็นรายปี





รูป 4.10 การกำหนดรายได้อื่นๆ

4) ส่วนรายได้อื่นๆให้ใส่เป็นรายปี

รูป 4.10 การกำหนดรายได้อื่นๆ

5) เพื่อจะดูคำนวนภาษีที่ต้องจ่าย ให้กดตรง **icon "ภาษี"**

รูป 4.11 คำนวนภาษี

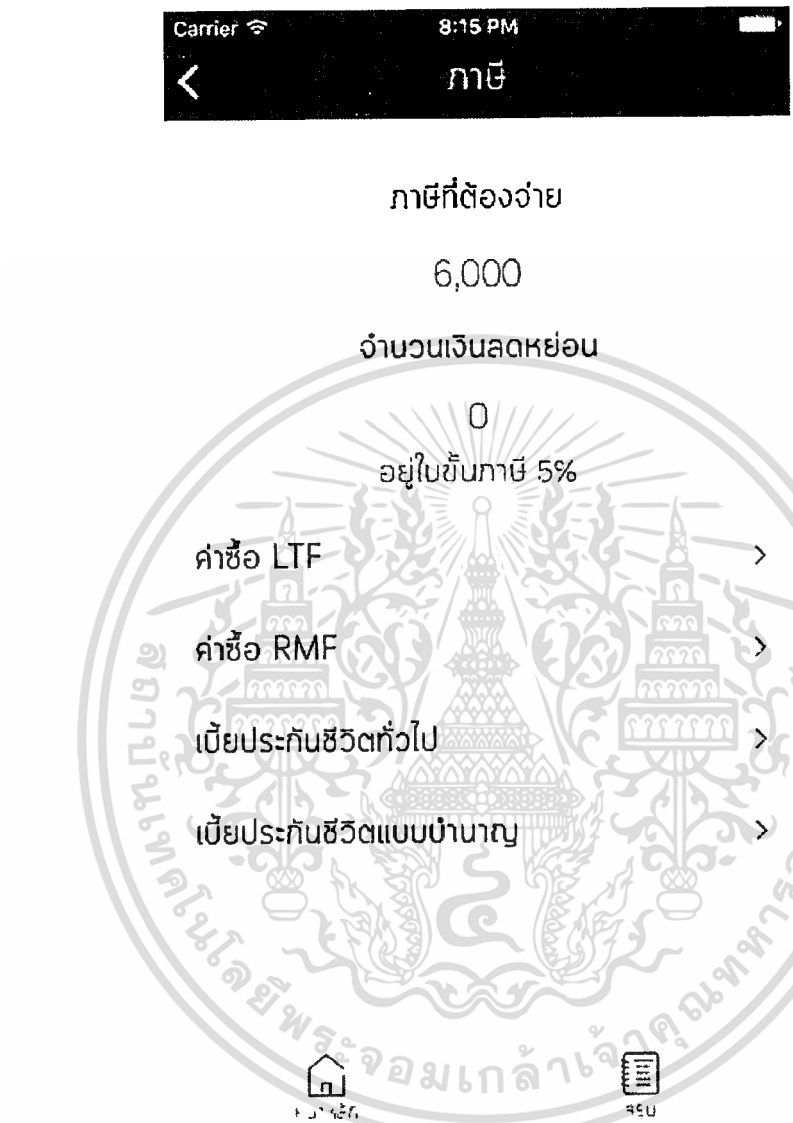
5) เพื่อจะดูคำนวนภาษีที่ต้องจ่าย ให้กดตรง **icon "ภาษี"**

รูป 4.11 คำนวนภาษี

6) เมื่อใส่รายได้เรียบร้อยจะขึ้นจำนวนภาษีที่ต้องจ่าย และสามารถใส่ค่าลดหย่อนต่างๆ ได้ โดยคำนวณจากเงินเดือนต่อปี - ค่าใช้จ่าย (40% ของรายได้แต่ไม่เกิน 60,000 บาท) - ค่าลดหย่อน

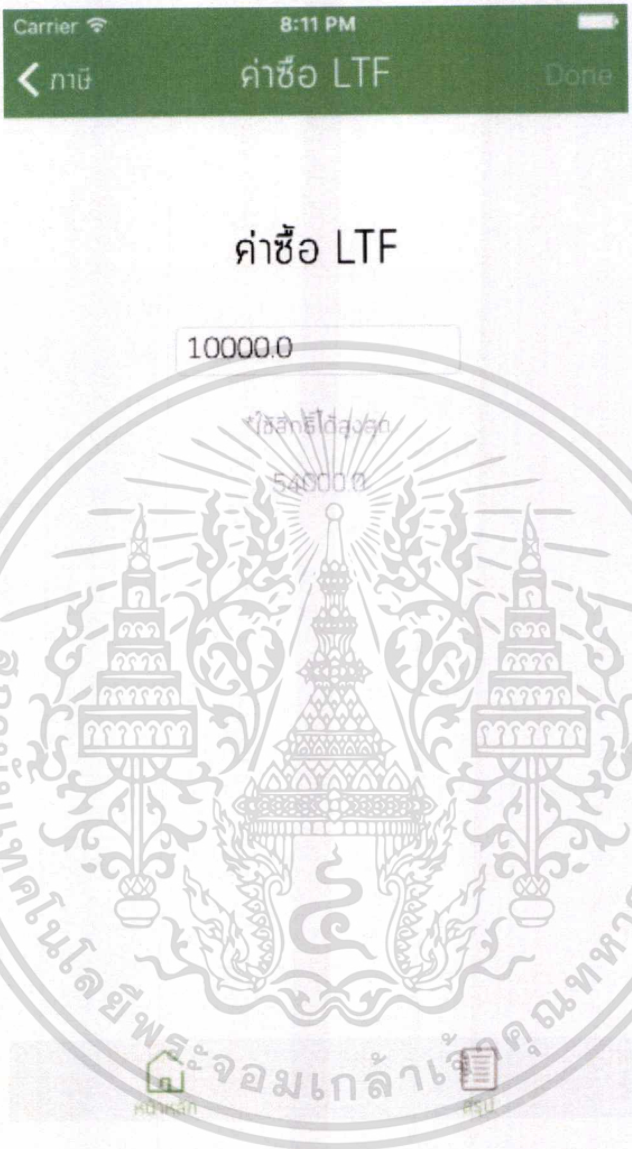
รูป 4.12 ภาษีที่ต้องจ่าย

6) เมื่อใส่รายได้เรียบร้อยจะขึ้นจำนวนภาษีที่ต้องจ่าย และสามารถใส่ค่าลดหย่อนต่างๆ ได้ โดยคำนวณจากเงินเดือนต่อปี - ค่าใช้จ่าย (40% ของรายได้แต่ไม่เกิน 60,000 บาท) - ค่าลดหย่อน



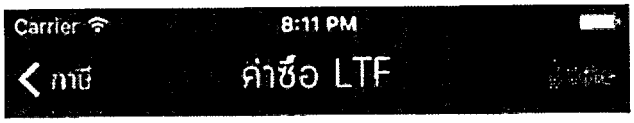
รูป 4.12 ภาษีที่ต้องจ่าย

7) ยกตัวอย่างเช่น ลดหย่อนค่า LTF สามารถซื้อสูงสุดที่ลดหย่อนได้ 54,000 บาท คิดจาก 15% ของรายได้



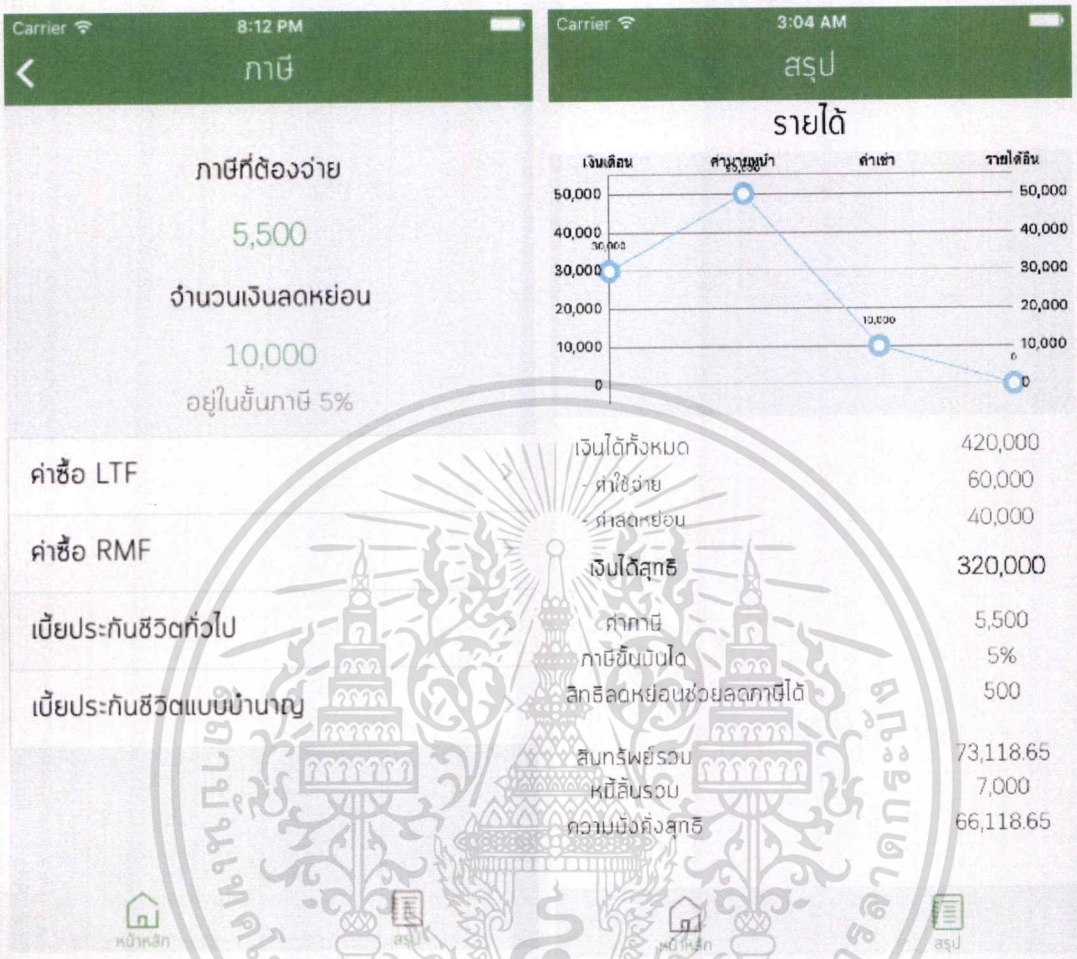
รูป 4.13 ลดหย่อน LTF

7) ยกตัวอย่างเช่น ลดหย่อนค่า LTF สามารถซื้อสูงสุดที่ลดหย่อนได้ 54,000 บาท คิดจาก 15% ของรายได้



รูป 4.13 ลดหย่อน LTF

8) เมื่อลดหย่อนแล้วจะได้ค่าภาษีที่จ่ายไปและแสดงจำนวนลดหย่อนและสามารถดูรายละเอียดได้ในหน้าสรุป



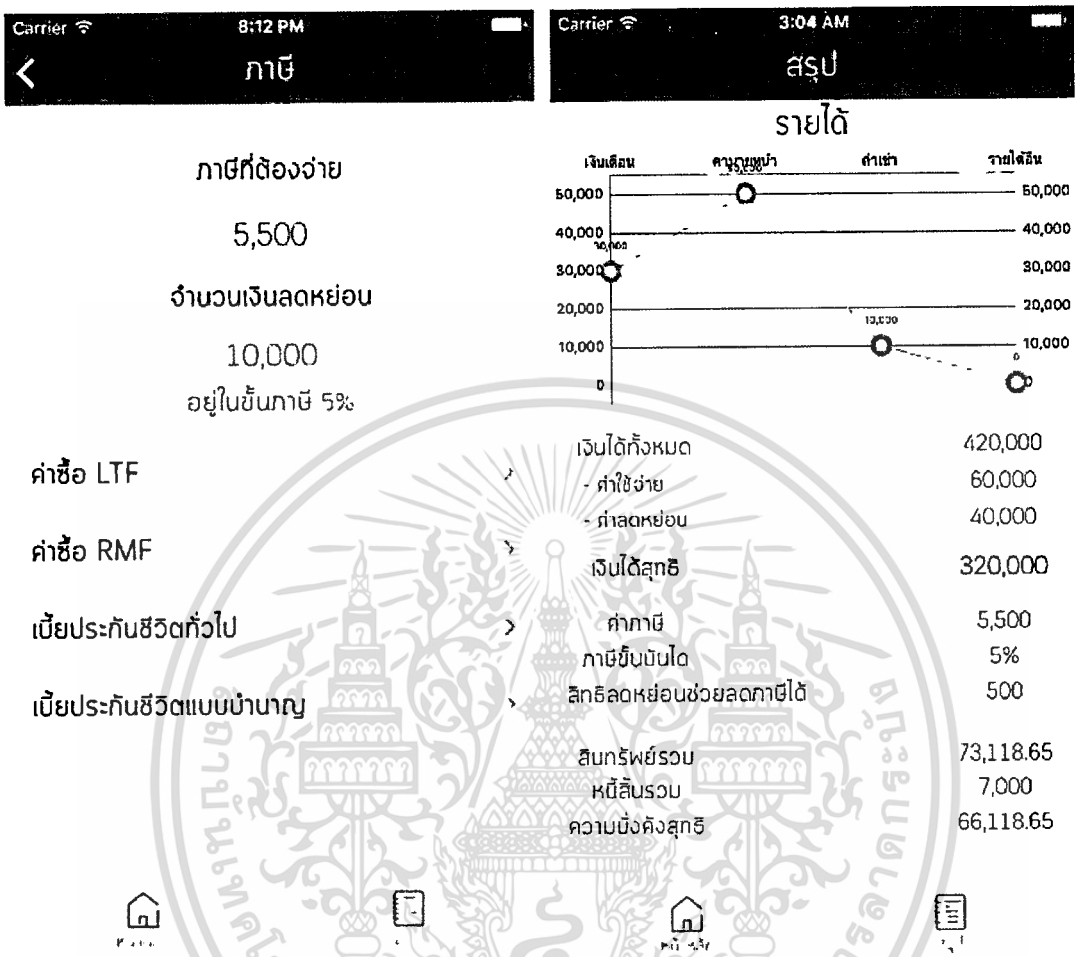
ก) ข)

รูป 4.14 หน้าจอภาษีและหน้าจอสรุป

ก) หน้าจอภาษี

ข) หน้าจอสรุป

8) เมื่อลดหย่อนแล้วจะได้ค่าภาษีที่จ่ายไปและแสดงจำนวนลดหย่อนและสามารถดูรายละเอียดได้ในหน้าสรุป



ก) ข)

รูป 4.14 หน้าจอภาษีและหน้าจอสรุป

ก) หน้าจอภาษี

ข) หน้าจอสรุป

4.3 ตรวจสุขภาพการเงิน

เป็นการเช็คสุขภาพการเงินโดยคำนวนจากงบการเงินและเทียบกับเกณฑ์มาตรฐานของนักวางแผนการเงิน

1) หน้าจอใส่รายละเอียดงบดุลของผู้ใช้เพื่อวิเคราะห์การเงิน

รูป 4.15 ตรวจสุขภาพการเงิน

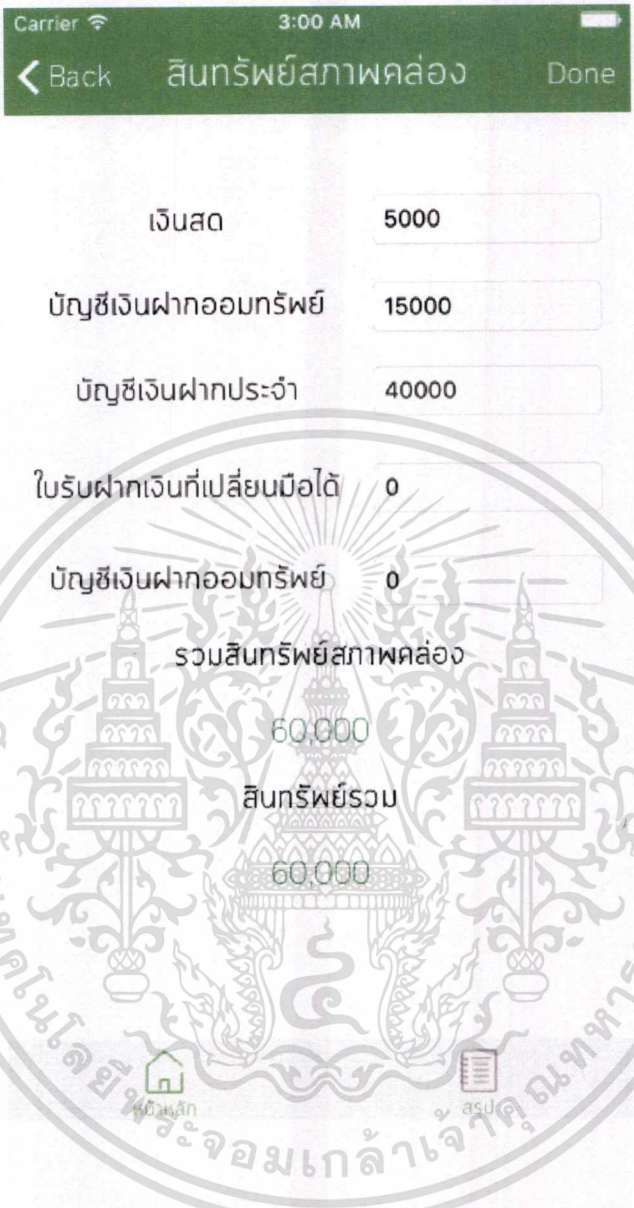
4.3 ตรวจสุขภาพการเงิน

เป็นการเช็คสุขภาพการเงินโดยคำนวนจากงบการเงินและเทียบกับเกณฑ์มาตรฐานของนักวางแผนการเงิน

1) หน้าจอใส่รายละเอียดงบดุลของผู้ใช้เพื่อวิเคราะห์การเงิน

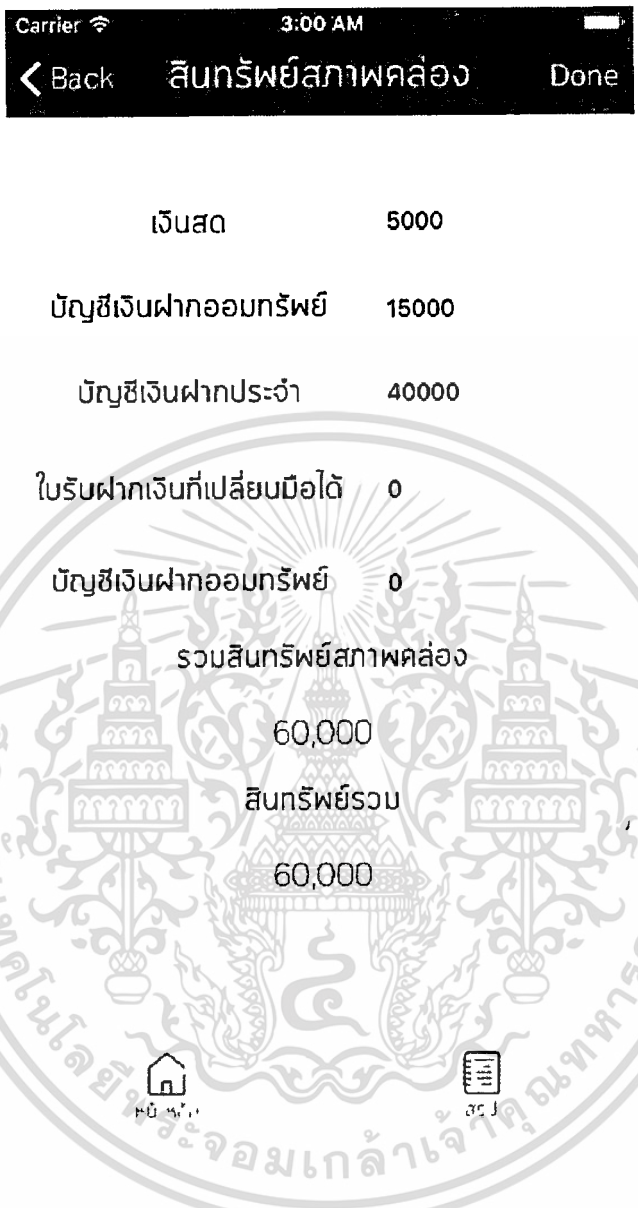
รูป 4.15 ตรวจสุขภาพการเงิน

2) ตัวอย่างสินทรัพย์สภาพคล่อง อย่าง เช่น เงินสด ออมทรัพย์



รูป 4.16 สินทรัพย์สภาพคล่อง

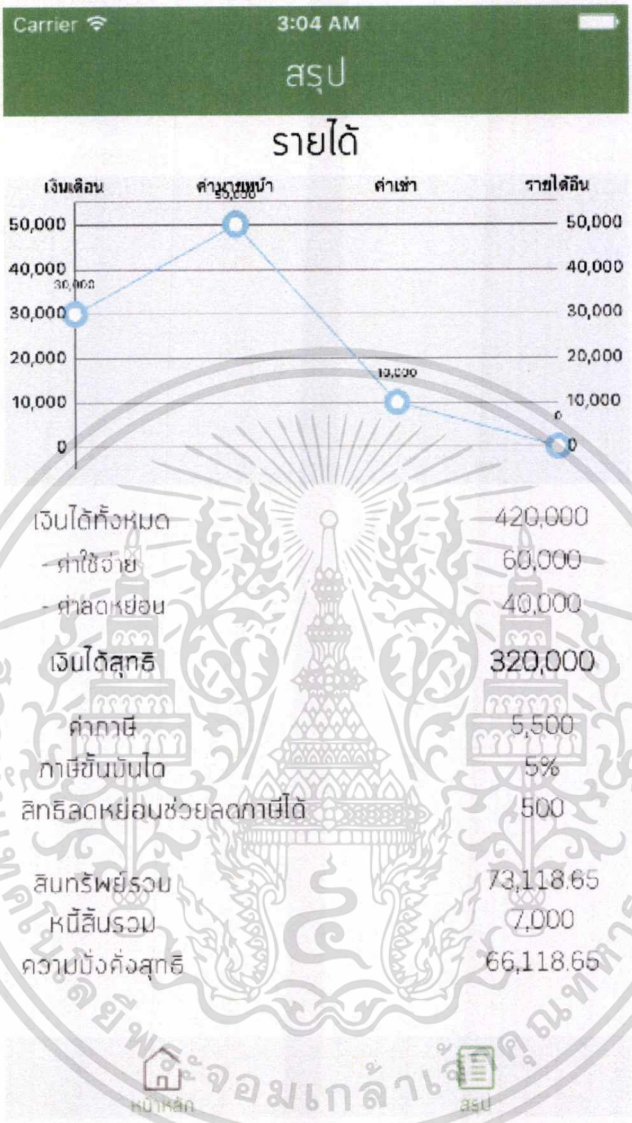
2) ตัวอย่างสินทรัพย์สภาพคล่อง อย่าง เช่น เงินสด ออมทรัพย์



รูป 4.16 สินทรัพย์สภาพคล่อง

4.4 หน้าจอสรุปผล

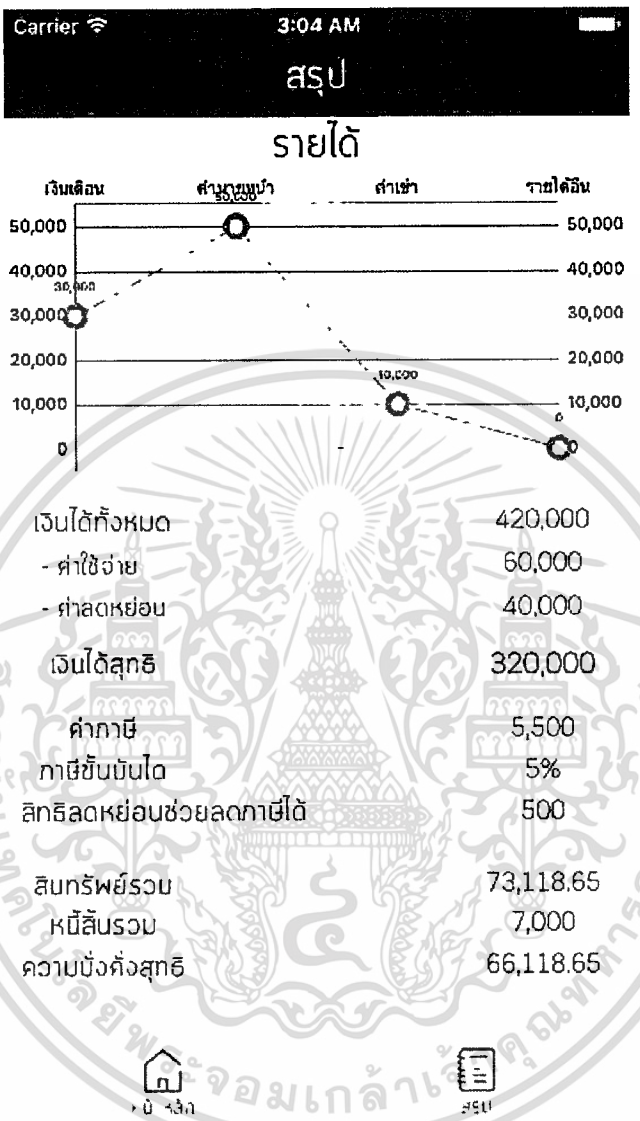
แสดงรายได้เป็นกราฟและสรุปส่วนของภาษีทั้งรายได้, ค่าลดหย่อน, อัตราภาษี



รูป 4.17 สรุปผลการใช้จ่าย

4.4 หน้าจอสรุปผล

แสดงรายได้เป็นกราฟและสรุปส่วนของภาษีทั้งรายได้, ค่าลดหย่อน, อัตราภาษี



รูป 4.17 สรุปผลการใช้จ่าย

4.5 ยกตัวอย่างการใช้งาน

4.5.1 ยกตัวอย่างการตั้งเป้าหมายการเงิน

นายสมคิด ต้องการวางแผนเก็บเงินเพื่อดาวน์บ้าน จำนวนเงินดาวน์เป็นจำนวนเงิน 100,000 บาท ต้องการเก็บระยะเวลา 5 ปี โดยเก็บในกองทุนตราสารหนี้ผลตอบแทนประมาณ 3 % นายสมคิดต้องวางแผนเก็บออมต่อเดือนเป็นจำนวนเงินเท่าใด

ในแอพพลิเคชั่นให้ไปที่การตั้งเป้าหมายการเงินแล้วใส่ข้อมูลดังนี้

เป้าหมาย = เงินดาวน์บ้าน

จำนวนเงิน = 100,000 บาท

ระยะเวลา = 5 ปี

ดอกเบี้ย = 3%

โดยวิธีคำนวณอ้างอิงจากสมการ 2.1 เมื่อใส่ข้อมูลเรียบร้อยแล้ว จะได้จำนวนเงินที่ออมต่อเดือนออกมา 1,569.62 บาท

ก) ข)

รูป 4.18 ตัวอย่างการใช้งานเป้าหมายทางการเงิน

ก) หน้าจอใส่ข้อมูลเป้าหมายการเงิน

ข) จำนวนเงินที่ออมต่อเดือน

4.5 ยกตัวอย่างการใช้งาน

4.5.1 ยกตัวอย่างการตั้งเป้าหมายการเงิน

นายสมคิด ต้องการวางแผนเก็บเงินเพื่อดาวน์บ้าน จำนวนเงินดาวน์เป็นจำนวนเงิน 100,000 บาท ต้องการเก็บระยะเวลา 5 ปี โดยเก็บในกองทุนตราสารหนี้ผลตอบแทนประมาณ 3 % นายสมคิดต้องวางแผนเก็บออมต่อเดือนเป็นจำนวนเงินเท่าใด

ในแอพพลิเคชั่นให้ไปที่การตั้งเป้าหมายการเงินแล้วใส่ข้อมูลดังนี้

เป้าหมาย = เงินดาวน์บ้าน

จำนวนเงิน = 100,000 บาท

ระยะเวลา = 5 ปี

ดอกเบี้ย = 3%

โดยวิธีคำนวณอ้างอิงจากสมการ 2.1 เมื่อใส่ข้อมูลเรียบร้อยแล้ว จะได้จำนวนเงินที่ออมต่อเดือนออกมา 1,569.62 บาท

ก) ข)

รูป 4.18 ตัวอย่างการใช้งานเป้าหมายทางการเงิน

ก) หน้าจอใส่ข้อมูลเป้าหมายการเงิน

ข) จำนวนเงินที่ออมต่อเดือน

4.5.2 ยกตัวอย่างการคำนวนภาษี

นายสมคิดมีเงินเดือน 30,000 บาท และซื้อ LTF จำนวน 10,000 บาท นายสมคิดจะเสียภาษีปีละเท่าใด

โดยวิธีคำนวณภาษีคิดจากเงินเดือนต่อปี - ค่าใช้จ่าย (40% ของรายได้แต่ไม่เกิน 60,000 บาท) - ค่าลดหย่อน จะได้จำนวนเงินที่เสียภาษีออกมาที่ 5,500 บาท โดยมีค่าลดหย่อน 10,000 บาทและอยู่ในขั้นภาษีที่ 5 %

ก) ข)

รูป 4.19 ตัวอย่างการใช้งานภาษี

ก) หน้าจอใส่ข้อมูลรายได้

ข) จำนวนเงินที่เสียภาษี

4.5.2 ยกตัวอย่างการคำนวนภาษี

นายสมคิดมีเงินเดือน 30,000 บาท และซื้อ LTF จำนวน 10,000 บาท นายสมคิดจะเสียภาษีปีละเท่าใด

โดยวิธีคำนวณภาษีคิดจากเงินเดือนต่อปี - ค่าใช้จ่าย (40% ของรายได้แต่ไม่เกิน 60,000 บาท) - ค่าลดหย่อน จะได้จำนวนเงินที่เสียภาษีออกมาที่ 5,500 บาท โดยมีค่าลดหย่อน 10,000 บาทและอยู่ในขั้นภาษีที่ 5 %

ก) ข)

รูป 4.19 ตัวอย่างการใช้งานภาษี

ก) หน้าจอใส่ข้อมูลรายได้

ข) จำนวนเงินที่เสียภาษี

4.5.3 ยกตัวอย่างการตรวจสุขภาพการเงิน

นายสมศักดิ์มีสินทรัพย์สภาพคล่อง 60,000 บาท, สินทรัพย์ลงทุน 3,118.65 บาท, สินทรัพย์ใช้ส่วนตัว 10,000 บาท, รวมกันเป็น 73,118.65 บาท

โดยที่มีหนี้สิ้นระยะสั้น 3,000 บาท, หนี้สั้นระยะยาว 4,000 บาท รวมกันเป็น 7,000 บาท นายสมคิดจะมีความมั่งคั่งสุทธิที่เท่าใด

ก) ข)

รูป 4.20 ตัวอย่างการใช้งานตรวจสุขภาพการเงิน

ก) หน้าจอใส่ข้อมูลสินทรัพย์สภาพคล่อง

ข) หน้าจอใส่ข้อมูลสินทรัพย์ลงทุน

4.5.3 ยกตัวอย่างการตรวจสุขภาพการเงิน

นายสมศักดิ์มีสินทรัพย์สภาพคล่อง 60,000 บาท, สินทรัพย์ลงทุน 3,118.65 บาท, สินทรัพย์ใช้ส่วนตัว 10,000 บาท, รวมกันเป็น 73,118.65 บาท

โดยที่มีหนี้สิ้นระยะสั้น 3,000 บาท, หนี้สั้นระยะยาว 4,000 บาท รวมกันเป็น 7,000 บาท นายสมคิดจะมีความมั่งคั่งสุทธิที่เท่าใด

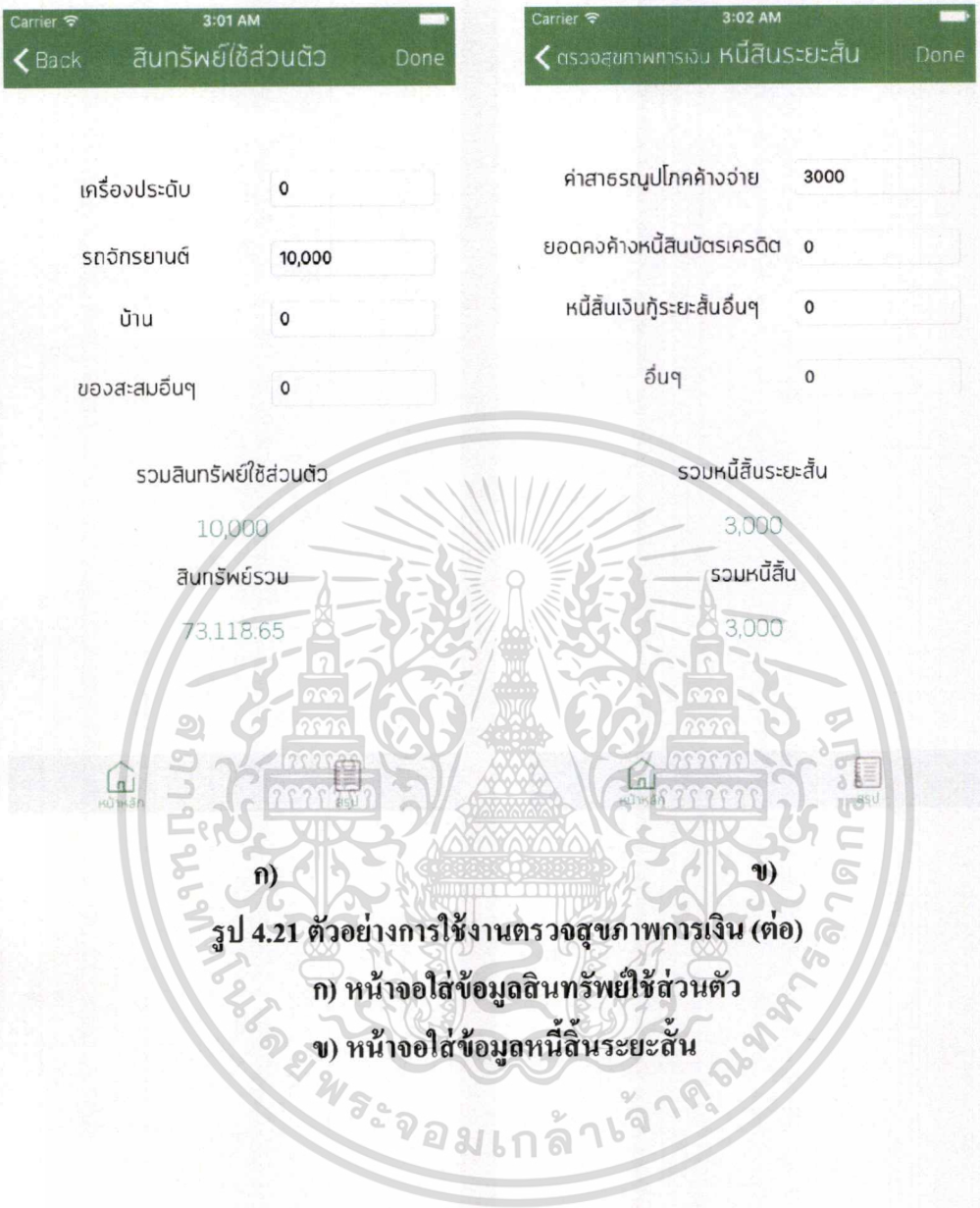
ก) ข)

รูป 4.20 ตัวอย่างการใช้งานตรวจสุขภาพการเงิน

ก) หน้าจอใส่ข้อมูลสินทรัพย์สภาพคล่อง

ข) หน้าจอใส่ข้อมูลสินทรัพย์ลงทุน

ตัวอย่างการกรอกข้อมูลสินทรัพย์ใช้ส่วนตัว, สินทรัพย์หนี้สิ้นระยะสั้น



ก) **ข)**

รูป 4.21 ตัวอย่างการใช้งานตรวจสุขภาพการเงิน (ต่อ)

ก) หน้าจอใส่ข้อมูลสินทรัพย์ใช้ส่วนตัว

ข) หน้าจอใส่ข้อมูลหนี้สิ้นระยะสั้น

ตัวอย่างการกรอกข้อมูลสินทรัพย์ใช้ส่วนตัว, สินทรัพย์หนี้สิ้นระยะสั้น

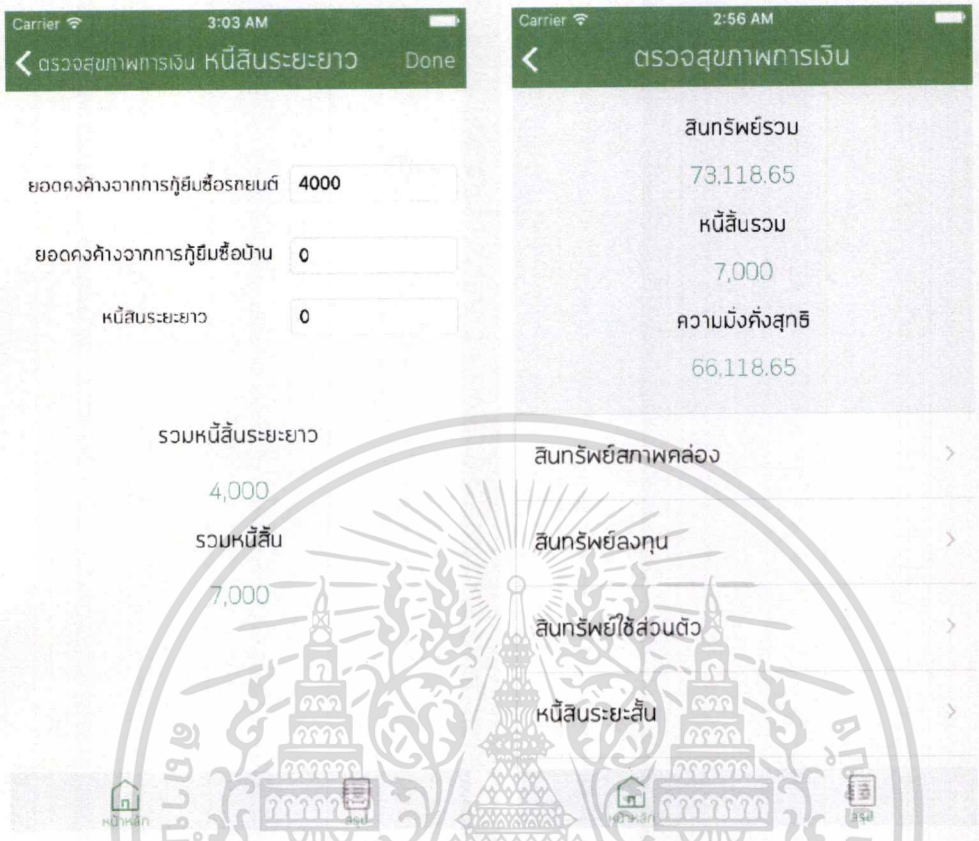
ก) ข)

รูป 4.21 ตัวอย่างการใช้งานตรวจสุขภาพการเงิน (ต่อ)

ก) หน้าจอใส่ข้อมูลสินทรัพย์ใช้ส่วนตัว

ข) หน้าจอใส่ข้อมูลหนี้สิ้นระยะสั้น

ตัวอย่างการกรอกข้อมูลหนี้สิ้นระยะยาว และดูผลลัพธ์รวมทั้งหมด



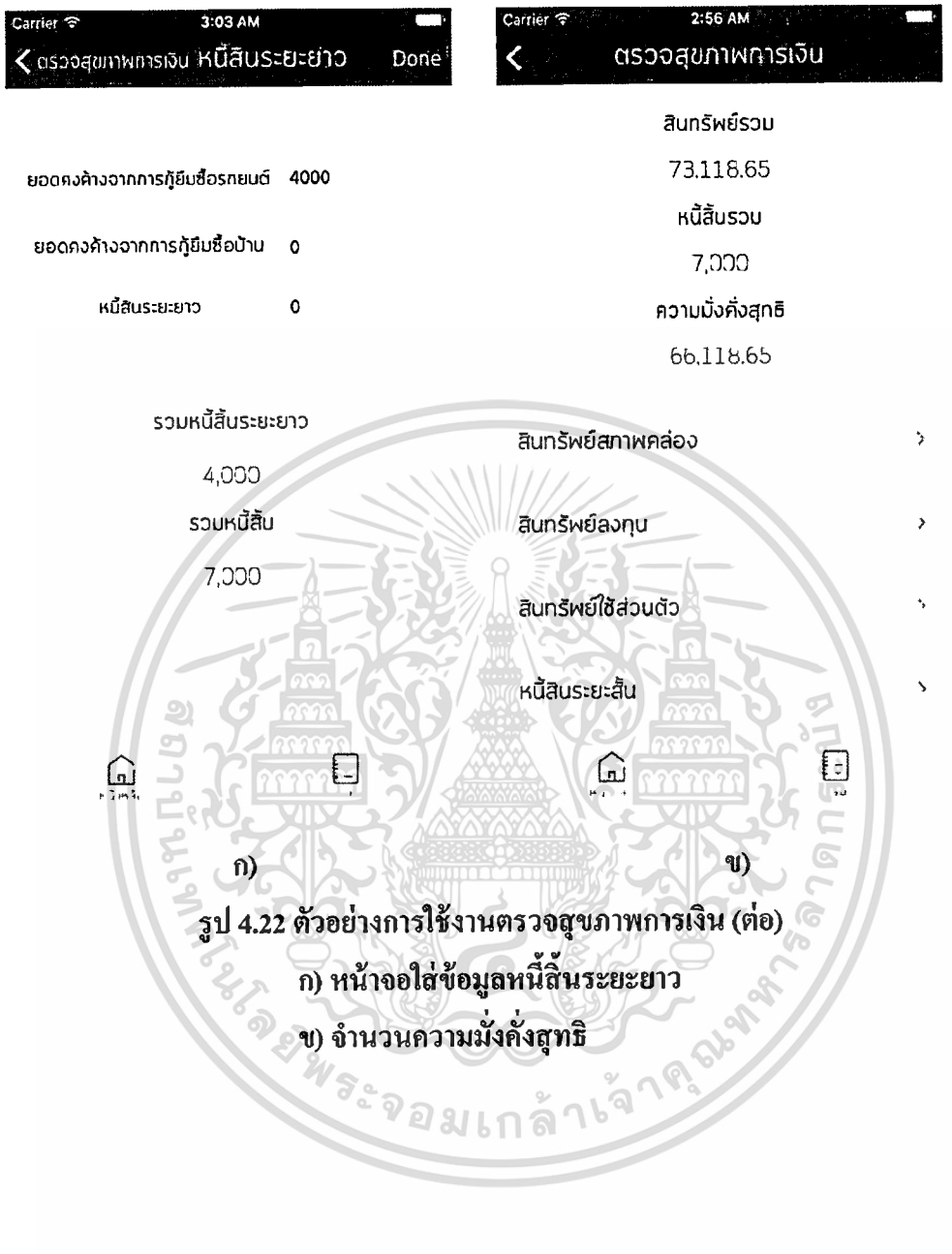
ก) **ข)**

รูป 4.22 ตัวอย่างการใช้งานตรวจสุขภาพการเงิน (ต่อ)

ก) หน้าจอใส่ข้อมูลหนี้สิ้นระยะยาว

ข) จำนวนความมั่งคั่งสุทธิ

ตัวอย่างการกรอกข้อมูลหนี้สิ้นระยะยาว และดูผลลัพธ์รวมทั้งหมด



ก) ข)

รูป 4.22 ตัวอย่างการใช้งานตรวจสุขภาพการเงิน (ต่อ)

ก) หน้าจอใส่ข้อมูลหนี้สิ้นระยะยาว

ข) จำนวนความมั่งคั่งสุทธิ

บทที่ 5
สรุปผลและข้อเสนอแนะ

5.1 สรุปผลจากโครงงาน

ในการศึกษาและพัฒนาแอพพลิเคชั่นการเงินส่วนบุคคลนั้นได้ข้อสรุปดังนี้ เราใช้หลักการเงินส่วนบุคคลในการประยุกต์ใช้กับแอพพลิเคชั่นการเงินส่วนบุคคลโดยใช้ข้อมูลทรัพย์สินและหนี้สินในการคำนวณตรวจสุขภาพการเงิน และใช้สูตรคำนวณในการวางแผนเป้าหมายการเงินเพื่อออกมาเป็นเป้าหมายที่ออมเงินต่อเดือนได้ รวมถึงคำนวณลดหย่อนสูงสุดที่สามารถลดหย่อน LTF,RMF และเบี้ยประกันชีวิตได้ เพื่อให้ผู้ใช้งานสะดวกในการวางแผนการเงินได้

5.1.1 เป้าหมายทางการเงิน

1) สามารถคำนวณหาจำนวนเงินให้ออมต่อเดือนได้
2) สามารถแสดงผลจำนวนเงินที่ออมต่อเดือน

5.1.2 ตรวจสุขภาพการเงิน

1) สามารถตรวจสุขภาพการเงินของเราได้
2) มีกราฟในการตรวจสุขภาพการเงิน

5.1.3 คำนวณการลดหย่อน

1) สามารถคำนวณจำนวนลดหย่อนสูงสุดของ LTF
2) สามารถคำนวณจำนวนลดหย่อนสูงสุดของ RMF
3) สามารถคำนวณจำนวนลดหย่อนสูงสุดของเบี้ยประกันชีวิตบำนาญ

5.2 ปัญหาอุปสรรคและแนวทางแก้ไข

1) เวลาใช้ Realm Database ทุกครั้งที่มีการ Write ต้องใส่ Transaction เสมอ
2) คลังข้อมูล มีคำถามที่ใช้ถามอย่างหลากหลายและโดยส่วนใหญ่มักจะเป็นคำถามทางด้านธุรกิจ ดังนั้นทำให้การพัฒนาโปรแกรมประยุกต์ที่ใช้สร้างคลังข้อมูลในโครงงานนี้ทำได้ค่อน ข้างยาก เพราะไม่ค่อยมีความรู้และความเข้าใจในทางธุรกิจพอสมควร จึงแก้ไขโดยการคัดเฉพาะคำถามที่สำคัญและพบบ่อยให้ผู้ใช้งานเลือกใช้ โดยกำหนดคำถามโดยผู้พัฒนาโปรแกรมประยุกต์
3) การสร้างสภาพแวดล้อมของฐานข้อมูลในการทดลองและศึกษาโครงงานนี้ค่อนข้างยาก และจำเป็นต้องใช้ข้อมูลที่เป็นข้อมูลตามสภาพเวลาจริง จึงมีแนวทางป้องกันข้อผิดพลาด

บทที่ 5
สรุปผลและข้อเสนอแนะ

5.1 สรุปผลจากโครงงาน

ในการศึกษาและพัฒนาแอพพลิเคชั่นการเงินส่วนบุคคลนั้นได้ข้อสรุปดังนี้ เราใช้หลักการเงินส่วนบุคคลในการประยุกต์ใช้กับแอพพลิเคชั่นการเงินส่วนบุคคลโดยใช้ข้อมูลทรัพย์สินและหนี้สินในการคำนวณตรวจสุขภาพการเงิน และใช้สูตรคำนวณในการวางแผนเป้าหมายการเงินเพื่อออกมาเป็นเป้าหมายที่ออมเงินต่อเดือนได้ รวมถึงคำนวณลดหย่อนสูงสุดที่สามารถลดหย่อน LTF,RMF และเบี้ยประกันชีวิตได้ เพื่อให้ผู้ใช้งานสะดวกในการวางแผนการเงินได้

5.1.1 เป้าหมายทางการเงิน

1) สามารถคำนวณหาจำนวนเงินให้ออมต่อเดือนได้
2) สามารถแสดงผลจำนวนเงินที่ออมต่อเดือน

5.1.2 ตรวจสุขภาพการเงิน

1) สามารถตรวจสุขภาพการเงินของเราได้
2) มีกราฟในการตรวจสุขภาพการเงิน

5.1.3 คำนวณการลดหย่อน

1) สามารถคำนวณจำนวนลดหย่อนสูงสุดของ LTF
2) สามารถคำนวณจำนวนลดหย่อนสูงสุดของ RMF
3) สามารถคำนวณจำนวนลดหย่อนสูงสุดของเบี้ยประกันชีวิตบำนาญ

5.2 ปัญหาอุปสรรคและแนวทางแก้ไข

1) เวลาใช้ Realm Database ทุกครั้งที่มีการ Write ต้องใส่ Transaction เสมอ
2) คลังข้อมูล มีคำถามที่ใช้ถามอย่างหลากหลายและโดยส่วนใหญ่มักจะเป็นคำถามทางด้านธุรกิจ ดังนั้นทำให้การพัฒนาโปรแกรมประยุกต์ที่ใช้สร้างคลังข้อมูลในโครงงานนี้ทำได้ค่อน ข้างยาก เพราะไม่ค่อยมีความรู้และความเข้าใจในทางธุรกิจพอสมควร จึงแก้ไขโดยการคัดเฉพาะคำถามที่สำคัญและพบบ่อยให้ผู้ใช้งานเลือกใช้ โดยกำหนดคำถามโดยผู้พัฒนาโปรแกรมประยุกต์
3) การสร้างสภาพแวดล้อมของฐานข้อมูลในการทดลองและศึกษาโครงงานนี้ค่อนข้างยาก และจำเป็นต้องใช้ข้อมูลที่เป็นข้อมูลตามสภาพเวลาจริง จึงมีแนวทางป้องกันข้อผิดพลาด

บรรณานุกรม

ศูนย์ส่งเสริมการพัฒนาความรู้ตลาดทุน, สถาบันกองทุนเพื่อพัฒนาตลาดทุน และตลาดหลักทรัพย์แห่งประเทศไทย. หลักสูตรการวางแผนการเงิน ชุดวิชาที่ 1 พื้นฐานการวางแผนการเงิน.

กรุงเทพฯ: ตลาดหลักทรัพย์

ศูนย์ส่งเสริมการพัฒนาความรู้ตลาดทุน, สถาบันกองทุนเพื่อพัฒนาตลาดทุน และตลาดหลักทรัพย์แห่งประเทศไทย. การบริหารการเงินส่วนบุคคล.
กรุงเทพฯ: ตลาดหลักทรัพย์

ธนิดา จิตร์น้อมรัตน์. กรณีศึกษาการตรวจสุขภาพทางการเงินส่วนบุคคลด้วยอัตราส่วนทางการเงิน.
กรุงเทพฯ: มหาวิทยาลัยธุรกิจบัณฑิตย์

ฌลาวัลคุ์ ตรงชาติ และฉัฐธิดา คงสกุล . 2557. การพัฒนาแอพพลิเคชันบริจาคโลหิตบนระบบปฏิบัติการแอนดรอยด์". วิยานิพนธ์วิศวกรรมศาสตร์บัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

A-academy. **เริ่มต้นจากหลักสูตร Financial Foundation**. [Online]. Available: http://www.a-academy.net/start-here/

yuttana.me. **[iOS Tips]Cocoa Pods ทำความรู้จักกับ Library Dependency.** [Online]. Available: http://yuttana.me/2013/11/04/cocoa-pods/

Google Developers. **The best of Google Maps for every iOS app.** [Online]. Available : https://developers.google.com/maps/documentation/ios-sdk/

Apple Developer. **iOS Developer Library.** [Online]. Available: https://developer.apple.com/library/ios/navigation/

บรรณานุกรม

ศูนย์ส่งเสริมการพัฒนาความรู้ตลาดทุน, สถาบันกองทุนเพื่อพัฒนาตลาดทุน และตลาดหลักทรัพย์แห่งประเทศไทย. หลักสูตรการวางแผนการเงิน ชุดวิชาที่ 1 พื้นฐานการวางแผนการเงิน.

กรุงเทพฯ: ตลาดหลักทรัพย์

ศูนย์ส่งเสริมการพัฒนาความรู้ตลาดทุน, สถาบันกองทุนเพื่อพัฒนาตลาดทุน และตลาดหลักทรัพย์แห่งประเทศไทย. การบริหารการเงินส่วนบุคคล.
กรุงเทพฯ: ตลาดหลักทรัพย์

ธนิดา จิตร์น้อมรัตน์. กรณีศึกษาการตรวจสุขภาพทางการเงินส่วนบุคคลด้วยอัตราส่วนทางการเงิน.
กรุงเทพฯ: มหาวิทยาลัยธุรกิจบัณฑิตย์

ฌลาวัลคุ์ ตรงชาติ และณัฐธิดา คงสกุล . 2557. การพัฒนาแอพพลิเคชันบริจาคโลหิตบนระบบปฏิบัติการแอนดรอยด์". วิยานิพนธ์วิศวกรรมศาสตร์บัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

A-academy. **เริ่มต้นจากหลักสูตร Financial Foundation**. [Online]. Available: http://www.a-academy.net/start-here/

yuttana.me. **[iOS Tips]Cocoa Pods ทำความรู้จักกับ Library Dependency.** [Online]. Available: http://yuttana.me/2013/11/04/cocoa-pods/

Google Developers. **The best of Google Maps for every iOS app.** [Online]. Available : https://developers.google.com/maps/documentation/ios-sdk/

Apple Developer. **iOS Developer Library.** [Online]. Available: https://developer.apple.com/library/ios/navigation/

Mac Thai. **รู้จักกับ Swift ภาษาที่ทำให้การเขียนแอพบน iOS, OS X ง่ายนิดเดียว. [Online].** **Available:** http://www.macthai.com/2014/06/07/intruduction-to-swift-programming-language-from-apple/

Appcodev. **พื้นฐานการเขียนโปรแกรมภาษา Swift.** [Online]. Available: http://www.appcodev.com/swift-พื้นฐานการเขียนโปรแกรม

NuuNeoI. **เปิดสู่โลกใหม่ "Swift" แห่งอาณาจักร Apple ฉบับเบื้องต้นสำหรับนักพัฒนา.** [Online]. Available: http://nuuneoi.com/blog/blog.php?read_id=684

TaxBugnom. **คู่มือลดภาษี สั้น ง่าย ใช้ได้ทันที.** [Online]. Available: http://tax.bugnoms.com/book-store/

Mac Thai. **รู้จักกับ Swift ภาษาที่ทำให้การเขียนแอพบน iOS, OS X ง่ายนิดเดียว. [Online].** **Available:** http://www.macthai.com/2014/06/07/intruduction-to-swift-programming-language-from-apple/

Appcodev. **พื้นฐานการเขียนโปรแกรมภาษา Swift.** [Online]. Available: http://www.appcodev.com/swift-พื้นฐานการเขียนโปรแกรม

NuuNeoI. **เปิดสู่โลกใหม่ "Swift" แห่งอาณาจักร Apple ฉบับเบื้องต้นสำหรับนักพัฒนา.** [Online]. Available: http://nuuneoi.com/blog/blog.php?read_id=684

TaxBugnom. **คู่มือลดภาษี สั้น ง่าย ใช้ได้ทันที.** [Online]. Available: http://tax.bugnoms.com/book-store/

ภาคผนวก ก

รายละเอียดยูสเคส

ยูสเคสคำนวณภาษี (Tax)

ชื่อยูสเคส (Use Case Name)	**Fund**
ตัวกระทำหลัก (Actor) :	ผู้ใช้งาน
รายละเอียด (Description) :	คำนวณภาษีใช้ในการลดหย่อน
เงื่อนไขก่อนสำเร็จ (Pre-Condition) :	ผู้ใช้งานกรอกข้อมูลรายได้ - รายจ่าย
เงื่อนไขหลังสำเร็จ (Post-Condition) :	แสดงจำนวนการลดหย่อนภาษี
ผังการทำงานหลัก (Main Flow) :	แสดงข้อมูลภาษี LTF, RMF สามารถลดหย่อนภาษีได้เท่าไร
ผังการยกเว้น (Exceptional Flow) :	-

ยูสเคสการตั้งเป้าหมาย (Finance Goal)

ชื่อยูสเคส (Use Case Name)	**Fund**
ตัวกระทำหลัก (Actor) :	ผู้ใช้งาน
รายละเอียด (Description) :	กำหนดเป้าหมายการเงิน
เงื่อนไขก่อนสำเร็จ (Pre-Condition) :	ผู้ใช้งานกรอกเป้าหมาย จำนวนเงิน และเวลา
เงื่อนไขหลังสำเร็จ (Post-Condition) :	แสดงจำนวนเงินที่ควรออมในแต่ละเดือน
ผังการทำงานหลัก (Main Flow) :	ผู้ใช้งานสามารถกำหนดเป้าหมายการเงินได้
ผังการยกเว้น (Exceptional Flow) :	-

ภาคผนวก ก

รายละเอียดยูสเคส

ยูสเคสคำนวณภาษี (Tax)

ชื่อยูสเคส (Use Case Name)	**Fund**
ตัวกระทำหลัก (Actor) :	ผู้ใช้งาน
รายละเอียด (Description) :	คำนวณภาษีใช้ในการลดหย่อน
เงื่อนไขก่อนสำเร็จ (Pre-Condition) :	ผู้ใช้งานกรอกข้อมูลรายได้ - รายจ่าย
เงื่อนไขหลังสำเร็จ (Post-Condition) :	แสดงจำนวนการลดหย่อนภาษี
ผังการทำงานหลัก (Main Flow) :	แสดงข้อมูลภาษี LTF, RMF สามารถลดหย่อนภาษีได้เท่าไร
ผังการยกเว้น (Exceptional Flow) :	-

ยูสเคสการตั้งเป้าหมาย (Finance Goal)

ชื่อยูสเคส (Use Case Name)	**Fund**
ตัวกระทำหลัก (Actor) :	ผู้ใช้งาน
รายละเอียด (Description) :	กำหนดเป้าหมายการเงิน
เงื่อนไขก่อนสำเร็จ (Pre-Condition) :	ผู้ใช้งานกรอกเป้าหมาย จำนวนเงิน และเวลา
เงื่อนไขหลังสำเร็จ (Post-Condition) :	แสดงจำนวนเงินที่ควรออมในแต่ละเดือน
ผังการทำงานหลัก (Main Flow) :	ผู้ใช้งานสามารถกำหนดเป้าหมายการเงินได้
ผังการยกเว้น (Exceptional Flow) :	-

ยูสเคสบันทึกตรวจสุขภาพการเงิน (Finace Health)

ชื่อยูสเคส (Use Case Name)	**Fund**
ตัวกระทำหลัก (Actor) :	ผู้ใช้งาน
รายละเอียด (Description) :	ข้อมูลสินทรัพย์ หนี้สิ้น
เงื่อนไขก่อนสำเร็จ (Pre-Condition) :	ผู้ใช้งานกรอกจำนวนสินทรัพย์ และหนี้สิ้น
เงื่อนไขหลังสำเร็จ (Post-Condition) :	แสดงผลลัพธ์ตามที่ผู้ใช้กรอก
ผังการทำงานหลัก (Main Flow) :	ผู้ใช้งานสามารถตรวจสอบสินทรัพย์ หนี้สิ้นได้
ผังการยกเว้น (Exceptional Flow) :	-

ยูสเคสรายได้ (Income)

ชื่อยูสเคส (Use Case Name)	**See Information**
ตัวกระทำหลัก (Actor) :	ผู้ใช้งาน
รายละเอียด (Description) :	รายได้ของผู้ใช้
เงื่อนไขก่อนสำเร็จ (Pre-Condition) :	-
เงื่อนไขหลังสำเร็จ (Post-Condition) :	ได้รับข้อมูลรายได้
ผังการทำงานหลัก (Main Flow) :	ข้อมูลรายได้ของผู้ใช้
ผังการยกเว้น (Exceptional Flow) :	-

ยูสเคสรายละเอียดทั้งหมด (Dashboard)

ชื่อยูสเคส (Use Case Name)	**Dashboard**
ตัวกระทำหลัก (Actor) :	ผู้ใช้งาน
รายละเอียด (Description) :	แสดงรายละเอียดทั้งหมดเป็นรูปแบบกราฟ

ยูสเคสบันทึกตรวจสุขภาพการเงิน (Finace Health)

ชื่อยูสเคส (Use Case Name)	**Fund**
ตัวกระทำหลัก (Actor) :	ผู้ใช้งาน
รายละเอียด (Description) :	ข้อมูลสินทรัพย์ หนี้สิ้น
เงื่อนไขก่อนสำเร็จ (Pre-Condition) :	ผู้ใช้งานกรอกจำนวนสินทรัพย์ และหนี้สิ้น
เงื่อนไขหลังสำเร็จ (Post-Condition) :	แสดงผลลัพธ์ตามที่ผู้ใช้กรอก
ผังการทำงานหลัก (Main Flow) :	ผู้ใช้งานสามารถตรวจสอบสินทรัพย์ หนี้สิ้นได้
ผังการยกเว้น (Exceptional Flow) :	-

ยูสเคสรายได้ (Income)

ชื่อยูสเคส (Use Case Name)	**See Information**
ตัวกระทำหลัก (Actor) :	ผู้ใช้งาน
รายละเอียด (Description) :	รายได้ของผู้ใช้
เงื่อนไขก่อนสำเร็จ (Pre-Condition) :	-
เงื่อนไขหลังสำเร็จ (Post-Condition) :	ได้รับข้อมูลรายได้
ผังการทำงานหลัก (Main Flow) :	ข้อมูลรายได้ของผู้ใช้
ผังการยกเว้น (Exceptional Flow) :	-

ยูสเคสรายละเอียดทั้งหมด (Dashboard)

ชื่อยูสเคส (Use Case Name)	**Dashboard**
ตัวกระทำหลัก (Actor) :	ผู้ใช้งาน
รายละเอียด (Description) :	แสดงรายละเอียดทั้งหมดเป็นรูปแบบกราฟ

ชื่อยูสเคส (Use Case Name)	**Dashboard**
เงื่อนไขก่อนสำเร็จ (Pre-Condition) :	กรอกข้อมูลในฟังก์ชั่นหลัก อย่างน้อย 1 ประเภท
เงื่อนไขหลังสำเร็จ (Post-Condition) :	**แสดงผลลัพธ์ในรูปแบบกราฟ**
ผังการทำงานหลัก (Main Flow) :	แสดงผลลัพธ์ในรูปแบบกราฟ แจ้งเตือนผู้ใช้งานว่าควรใช้เงินเท่าไร
ผังการยกเว้น (Exceptional Flow) :	**กรณียังไม่ป้อนค่า จะไม่แสดงหน้าอะไร**

ชื่อยูสเคส (Use Case Name)	**Dashboard**
เงื่อนไขก่อนสำเร็จ (Pre-Condition) :	กรอกข้อมูลในฟังก์ชั่นหลัก อย่างน้อย 1 ประเภท
เงื่อนไขหลังสำเร็จ (Post-Condition) :	แสดงผลลัพธ์ในรูปแบบกราฟ
ผังการทำงานหลัก (Main Flow) :	แสดงผลลัพธ์ในรูปแบบกราฟ แจ้งเตือนผู้ใช้งานว่าควรใช้เงินเท่าไร
ผังการยกเว้น (Exceptional Flow) :	กรณียังไม่ป้อนค่า จะไม่แสดงหน้าอะไร