

Dream

Money Logging

GitHub repository : Dream

ผู้พัฒนา

นายพีรวิชญ์ ประพันธ์วิทยา 54010944

นายอภิรัตน์ พุทธรักษา 54011491

Cloud Computing, ภาคการศึกษาที่ 1/2557

## บทคัดย่อ (Abstract)

ในปัจจุบันนี้ปฏิเสธไม่ได้เลยว่าเงินนั้นถือเป็นส่วนหนึ่งที่สำคัญที่ช่วยในการดำรงชีวิตหรือใช้ชีวิตให้ดียิ่งขึ้น แต่บุคคลส่วนใหญ่มักมองข้ามวิธีการหรือหลักการในการบริหารจัดการเงินที่ได้รับมา และจากที่ได้ทำการศึกษาในขั้นต้นพบว่าคนไทยส่วนใหญ่ในปัจจุบันไม่มีการออมเงินเป็นประจำอีกทั้งยังไม่มีการวางแผนทางการเงินอีกด้วย จึงทำให้เกิดปัญหาต่างๆในอนาคต ซึ่งโครงการนี้จัดทำขึ้นมาเพื่อช่วยในการแก้ปัญหาที่กล่าวมาในขั้นต้น โดยระบบนั้นสามารถให้ผู้ใช้ทำการบันทึกการทำธุรกรรมทางการเงินต่างๆ การวางแผนทางการเงิน อัตราดอกเบี้ยต่างๆ และช่วยให้ผู้ใช้บันทึกข้อมูลต่างๆได้ง่ายที่สุด จากนั้นระบบทำการรายงานผลสรุปการทำธุรกรรมให้แก่ผู้ใช้ทราบ ทำให้ผู้ใช้สามารถทราบได้ว่าในปัจจุบันตนเองมีความสามารถหรือสถานภาพทางการเงินมากน้อยเพียงใดและจะต้องทำการวางแผนการใช้จ่ายเงินในอนาคตต่อไปอย่างไรอีกด้วย

## ความเป็นมาและความสำคัญของปัญหา (Problem Statement)

เนื่องจากในปัจจุบันความมั่นคงทางการเงินนั้นเป็นสิ่งสำคัญของมนุษย์อย่างยิ่ง เนื่องจากการดำเนินชีวิตต่างๆล้วนมีความเกี่ยวข้องกับเงินทั้งสิ้น แต่บุคคลส่วนใหญ่มักมองข้ามวิธีการหรือหลักการบริหารจัดการเงินที่ได้รับมา จึงทำให้เกิดปัญหาต่างๆขึ้นในอนาคต เช่น ปัญหาหนี้สินที่เป็นปัญหาส่วนใหญ่ที่พบในปัจจุบัน ซึ่งส่งผลกระทบต่อความเป็นอยู่ของบุคคลหรือครอบครัว ผลที่ตามมาคือ ความเครียด ปัญหาครอบครัว เกิดอาชญากรรม การฆ่าตัวตาย ฯลฯ

จากปัญหาที่ได้กล่าวมานั้นจึงจำเป็นต้องมีระบบที่ช่วยในการบันทึกการทำธุรกรรมต่างๆ การวางแผนทางการเงิน เพื่อช่วยให้ผู้ใช้สามารถประเมินสถานภาพทางการเงินของตนเอง ตรวจสอบการทำธุรกรรมต่างๆและช่วยในวางแผนทางการเงินในอนาคตอีกด้วย

## แนวทางการพัฒนาและเทคนิคที่ใช้ (Proposed Approach)

การบันทึกการทำธุรกรรมทางการเงินนั้น ผู้ใช้ส่วนใหญ่จะจดบันทึกลงในสมุดหรือโปรแกรม Excel ซึ่งไม่สะดวกในการพกพาหรือการใช้งาน และเนื่องจากในปัจจุบันผู้ใช้สามารถเข้าถึงอินเทอร์เน็ตได้ตลอดเวลา ผู้จัดทำจึงจัดทำเว็บแอปพลิเคชันซึ่งใช้บริการ Azure: Microsoft's Cloud Platform ชนิด Web Sites ทำให้ผู้ใช้บันทึกการทำธุรกรรมได้ตลอดเวลาและสะดวกรวดเร็วยิ่งขึ้น โดยจะให้ผู้ใช้มีบัญชี (account) ส่วนตัวที่ทำหน้าที่ในการบันทึกข้อมูลต่างๆของผู้ใช้ จากนั้นระบบจะทำการคำนวณหรือประเมินสถานภาพทางการเงินของผู้ใช้งาน ในการพัฒนาเว็บแอปพลิเคชันนั้นผู้พัฒนาจะใช้ Spring Framework ,HTML, AngularJS และ SQL Database ในการเก็บข้อมูลต่างๆของผู้ใช้งาน โดยระบบจะสามารถรองรับการใช้งานจากผู้ใช้งานเป็นจำนวนมากและการขยายตัวในอนาคตได้

## งานที่เกี่ยวข้อง (Related works)

### Money Lover

เป็นแอปพลิเคชันในอุปกรณ์พกพา ซึ่งช่วยในการทำบันทึกการทำธุรกรรมต่างๆ การวางแผนทางการเงิน และเครื่องมือที่ช่วยในการจัดการค่าใช้จ่ายต่างๆ การคำนวณดอกเบี้ย หรือภาษี โดยแอปพลิเคชันจะให้ผู้ใช้มีบัญชีการใช้งานของตนเอง จากนั้นระบบจะทำการสำรองข้อมูลในระบบคลาวด์ซึ่งจะทำให้ผู้ใช้สามารถใช้การอุปกรณ์พกพาใดก็ได้ แต่ระบบนั้นไม่มีการประเมินสภาพทางการเงินแก่ผู้ใช้งาน เช่น สภาพคล่องทางการเงิน อัตราส่วนภาระหนี้สิน อัตราส่วนการออม เป็นต้น

ที่มา : [HTTPS://MONEYLOVER.ME/](https://moneylover.me/)

### บัญชีรายรับ รายจ่าย

เป็นแอปพลิเคชันที่ช่วยในการบันทึกรายรับรายจ่ายเท่านั้น โดยในการบันทึกรายรับรายจ่ายผู้ใช้สามารถเพิ่มรูปเพื่อแสดงรายละเอียดต่างๆลงไปข้อมูลรายรับรายจ่ายได้ และผู้ใช้สามารถสรุปรายรับรายจ่ายต่างๆได้ แต่ระบบยังไม่มีสำรองข้อมูลต่างๆไว้ในเครื่องแม่ข่าย ไม่มีระบบบัญชีที่ช่วยในด้านความปลอดภัยของผู้ใช้งาน และไม่สามารถประเมินสภาพทางการเงินได้ เช่น สภาพคล่องทางการเงิน อัตราส่วนภาระหนี้สิน อัตราส่วนการออม เป็นต้น

ที่มา : <https://play.google.com/store/apps/details?id=com.nice2studio.mymoney>

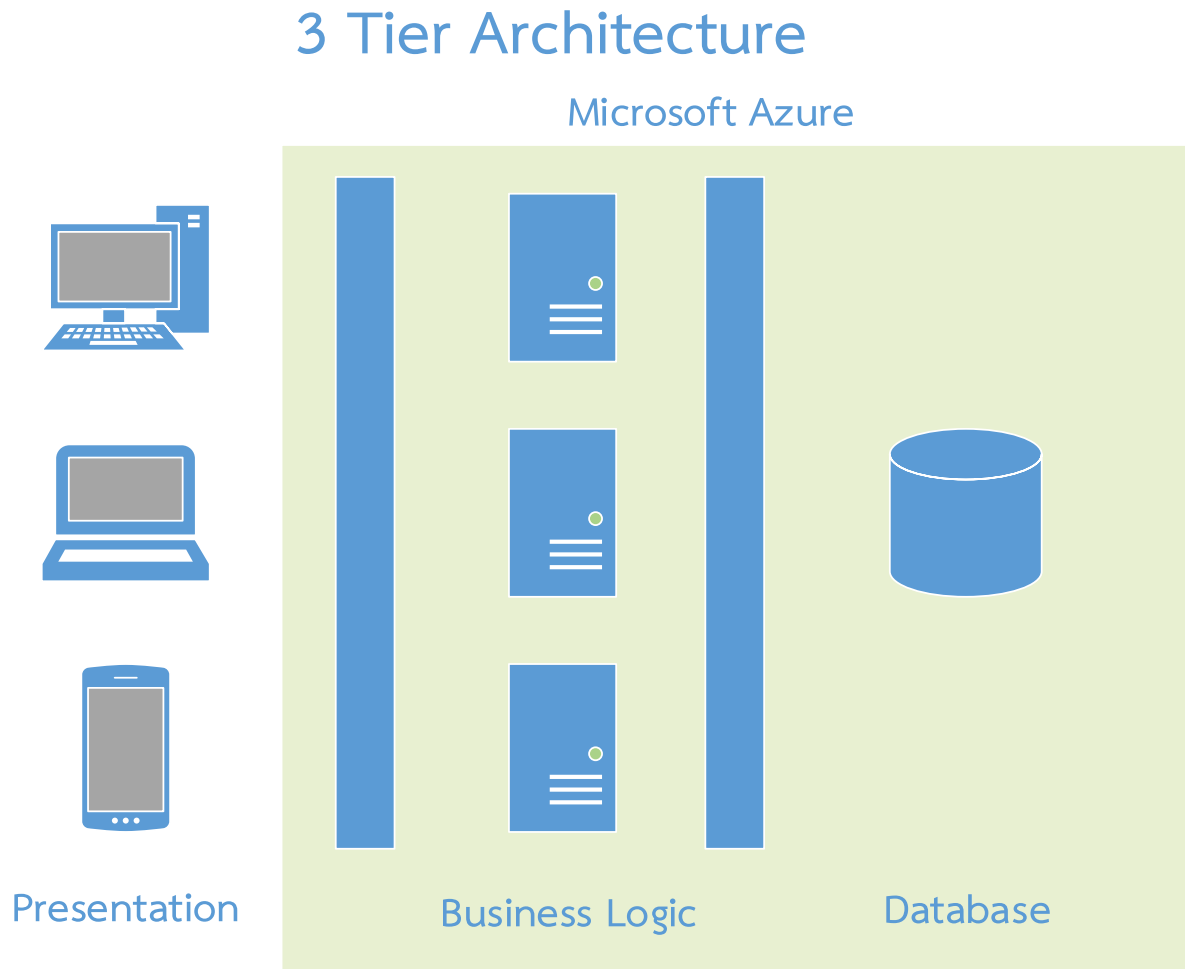
### การใช้งาน (Use Cases)

โครงการนี้ผู้ใช้สามารถบันทึกข้อมูลการทำธุรกรรมต่าง ตรวจสอบการทำธุรกรรมที่ผ่านมา ตรวจสอบสภาพทางการเงินในปัจจุบันของผู้ใช้งาน และเครื่องมือที่ช่วยในการวางแผนทางการเงินต่างๆ เช่น การตั้งเป้าหมายทางการเงิน การคำนวณอัตราดอกเบี้ยต่างๆ เป็นต้น

ตัวอย่างการคำนวณดอกเบี้ย ถ้าผู้ใช้ต้องการซื้อบ้านเป็นจำนวนเงิน 2 ล้านบาท ต้องการผ่อนเป็นจำนวน 20 ปี อัตราดอกเบี้ย 1.5% ผู้ใช้จะต้องทำการกรอกข้อมูลรายละเอียดต่างๆ จากนั้นระบบจะทำการคำนวณรายจ่ายต่อเดือนให้แก่ผู้ใช้งาน

## สถาปัตยกรรมของระบบ (System Design)

โครงสร้างของระบบโดยรวม



ภาพที่ 1 สถาปัตยกรรมของระบบ

โครงสร้างของระบบนั้นจะมีการแบ่งออกเป็น 3 ส่วนคือ

### 1.ส่วนที่ใช้ในการนำเสนอแก่ผู้ใช้งาน (Presentation)

ภายในส่วนนี้จะทำหน้าที่ในการนำเสนอข้อมูลต่างๆแก่ผู้ใช้งาน ซึ่งในการพัฒนานั้นจะใช้ HTTP , CSS , JavaScript ในการนำเสนอข้อมูลต่างๆผ่านทางเว็บเบราว์เซอร์ (Web Browser) โดยการแสดงผลนั้นจะออกแบบให้มีการปรับเปลี่ยนรูปแบบการใช้งานให้เป็นไปตามขนาดหน้าจอของอุปกรณ์ของผู้ใช้งาน (Responsive Web design)

## 2.ส่วนที่ใช้ในการประมวลผล (Business Logic)

ภายในส่วนนี้จะทำหน้าที่ในการประมวลผลข้อมูลและจัดการข้อมูลต่างๆของผู้ใช้งาน โดยประกอบไปด้วย

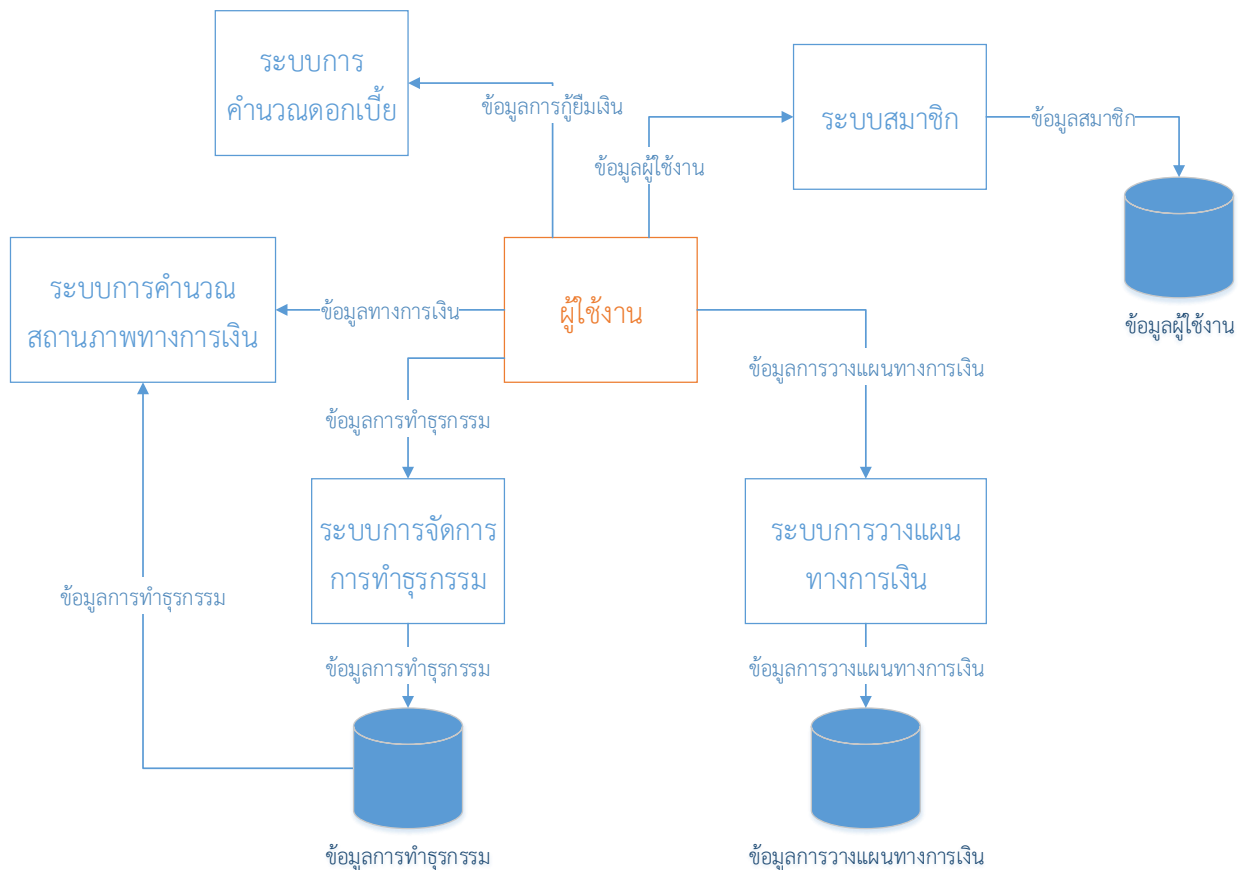
1. ระบบสมาชิก
2. ระบบการจัดการการทำธุรกรรม
3. ระบบการคำนวณสถานภาพทางการเงิน
4. ระบบการวางแผนทางการเงิน
5. ระบบการคำนวณดอกเบี้ย

ซึ่งพัฒนาขึ้นโดยใช้ภาษา Java และนำ Spring Framework มาช่วยในการควบคุมการทำงานให้อยู่ในรูปแบบ Model View Controller (MVC) โดยหลักการทำงานของ Framework นั้นจะมี Servlet ที่คอยทำหน้าที่ในการรับ Request จากผู้ใช้งานจากนั้นจะทำการส่งข้อมูลต่อไปยัง controller อื่นๆให้ทำงานต่อไป และระบบที่สร้างขึ้นนั้นจะใช้บริการ Website ของ Microsoft Azure และติดตั้ง Apache Tomcat ซึ่งเป็น web container ในการให้บริการ Java web application

## 3.ส่วนที่ใช้ในการเก็บข้อมูล (Database)

ภายในส่วนนี้จะทำหน้าที่ในการเก็บข้อมูลต่างๆของผู้ใช้งานที่ผ่านการประมวลผลจากส่วนประมวลผล (Business Logic) โดยจะเก็บข้อมูลอยู่ในฐานข้อมูล Relational Database ซึ่งจะใช้บริการ SQL database ของ Microsoft Azure

## โครงสร้างภายในของระบบ



ภาพที่ 2 โครงสร้างภายในของระบบ

การทำงานของระบบนั้นประกอบไปด้วย 5 ระบบย่อยคือ

### 1.ระบบสมาชิก

ระบบสมาชิคนั้นทำหน้าที่ในการจัดการข้อมูลรายละเอียดต่างๆของผู้ใช้งาน การเข้าสู่ระบบ การออกจากระบบ การตรวจสอบสถานะของผู้ใช้งาน และการควบคุมการเข้าถึงการทำงานของระบบในส่วนอื่นๆ เช่น ระบบการจัดการการทำธุรกรรมจะต้องเป็นสมาชิกเท่านั้นจึงจะสามารถใช้งานได้ เป็นต้น

### 2.ระบบการจัดการการทำธุรกรรม

ระบบการจัดการการทำธุรกรรมนั้นทำหน้าที่ในการจัดการข้อมูลการทำธุรกรรมต่างๆของสมาชิกในระบบเช่น การเพิ่ม/ลบข้อมูลการทำธุรกรรม การตรวจสอบการทำธุรกรรมที่ผ่านมาของสมาชิก เป็นต้น

### 3.ระบบการคำนวณสถานภาพทางการเงิน

ระบบการคำนวณสถานภาพทางการเงินนั้นทำหน้าที่ในการประเมินสถานภาพทางการเงินของผู้ใช้งานโดยไม่จำเป็นต้องสมัครสมาชิก ซึ่งถ้าเป็นผู้ใช้งานทั่วไประบบจะขอข้อมูลทางการเงินเพื่อใช้ในการคำนวณสภาพคล่องและสภาพคล่องพื้นฐานทางการเงินของผู้ใช้งาน แต่ถ้าผู้ใช้งานเป็นสมาชิกระบบจะใช้ข้อมูลจากข้อมูลการทำธุรกรรมทางการเงินของสมาชิกมาใช้ในการคำนวณ

### 4.ระบบการวางแผนทางการเงิน

ระบบการวางแผนทางการเงินนั้นทำหน้าที่ในการช่วยสมาชิกในการวางแผนทางการเงิน 3 ประเภทคือ การวางแผนงบประมาณ การวางแผนเงินออม การวางแผนเงินสำหรับการทำกิจกรรม โดยจะทำการรับข้อมูลจำนวนเงินเป้าหมายที่ต้องการ จำนวนเงินเริ่มต้น ระยะเวลาที่ต้องการวางแผน จากนั้นจะทำการสร้างบันทึกการวางแผนให้แก่ผู้ใช้งาน

### 5.ระบบการคำนวณดอกเบี้ย

ระบบการคำนวณดอกเบี้ยนั้นทำหน้าที่ในการคำนวณจำนวนเงินดอกเบี้ยจากการกู้ยืมเงินของผู้ใช้งาน โดยรับข้อมูลรายละเอียดของการกู้ยืมเงินและรายรับในแต่ละเดือน จากนั้นจะทำการคำนวณจำนวนเงินที่ต้องชำระและจำนวนเงินคงเหลือของผู้ใช้งานในแต่ละเดือนหรือในแต่ละรอบการชำระของผู้ใช้งาน

## รายละเอียดการพัฒนาซอฟต์แวร์

### ซอฟต์แวร์ที่เลือกใช้งาน

- Spring Framework และ Spring Security  
สำหรับการพัฒนา Web Service
- Apache Jmeter  
สำหรับการทดสอบประสิทธิภาพการทำงานของเว็บเซิร์ฟเวอร์
- Eclipse IDE for Java EE Developers  
สำหรับการพัฒนาเว็บแอปพลิเคชัน
- Apache Maven  
สำหรับการทำ Project Dependency
- Bower  
เป็น Project manager ของเว็บไซต์ในส่วน Presentation
- Angular material, AngularJS ,Bootstrap และ Polymer  
สำหรับการทำเว็บไซต์ในส่วน Presentation

### Cloud Service (Azure: Microsoft's Cloud Platform)

- Websites  
สำหรับการติดตั้งเว็บแอปพลิเคชันในส่วน Presentation และ Business logic
- SQL Database  
สำหรับการเก็บข้อมูลของผู้ใช้งาน



## Implementation Plan

กิจกรรม/เวลา	กันยายน				ตุลาคม				พฤศจิกายน			
	1	2	3	4	5	6	7	8	9	10	11	12
1.ศึกษาและออกแบบโครงสร้างของระบบ												
2.ออกแบบและสร้างฐานข้อมูล												
3.พัฒนาและทดสอบระบบสมาชิก												
4.พัฒนาและทดสอบระบบการจัดการการทำธุรกรรม												
5.พัฒนาและทดสอบระบบการวางแผนทางการเงิน												
6.พัฒนาและทดสอบระบบการคำนวณสถานภาพทางการเงิน												
7.พัฒนาและทดสอบระบบการคำนวณดอกเบี้ย												
8.ทดสอบการทำงานของระบบโดยรวม												
9.จัดทำเอกสารและส่งขึ้นงาน												

ตารางที่ 1 ตารางการวางแผนการทำงาน

## รายละเอียดการทำงาน

### 1.ศึกษาและออกแบบโครงสร้างของระบบ

ศึกษาการทำงานของ Microsoft Azure ในส่วน Website และ SQL database เพื่อนำมาใช้ในการออกแบบระบบให้สอดคล้องกับการทำงานและทำการศึกษาการใช้งาน Framework หรือ API ต่างๆที่เกี่ยวข้อง

ผู้รับผิดชอบ : นายพีรวิทย์ ประพันธ์วิทยา และ นายอภิรัตน์ พุทธิรักษา

### 2.ออกแบบและสร้างฐานข้อมูล

ทำการออกแบบและสร้างฐานข้อมูลเพื่อให้สอดคล้องกับการใช้งานภายในระบบ

ผู้รับผิดชอบ : นายพีรวิทย์ ประพันธ์วิทยา และ นายอภิรัตน์ พุทธิรักษา

### 3.พัฒนาและทดสอบระบบสมาชิก

ออกแบบและพัฒนาการจัดการสมาชิก ส่วนติดต่อผู้ใช้งาน การยืนยันตัวตน สิทธิในการเข้าใช้งานส่วนต่างๆของระบบ และทำการทดสอบการทำงานของระบบ

ผู้รับผิดชอบ : นายพีรวิทย์ ประพันธ์วิทยา

### 4.พัฒนาและทดสอบระบบการจัดการการทำธุรกรรม

ออกแบบและพัฒนาการจัดการการทำธุรกรรม (เพิ่ม/ลบ/ค้นหา) ส่วนติดต่อผู้ใช้งาน การคำนวณจำนวนเงินคงเหลือ และทำการทดสอบการทำงานของระบบ

ผู้รับผิดชอบ : นายอภิรัตน์ พุทธิรักษา

### 5.พัฒนาและทดสอบระบบการวางแผนทางการเงิน

ออกแบบและพัฒนาระบบการวางแผนทางการเงิน ส่วนติดต่อผู้ใช้งาน และทำการทดสอบการทำงานของระบบ

ผู้รับผิดชอบ : นายพีรวิทย์ ประพันธ์วิทยา

### 6.พัฒนาและทดสอบระบบการคำนวณสถานภาพทางการเงิน

ออกแบบและพัฒนาการคำนวณสถานภาพทางการเงิน ส่วนติดต่อผู้ใช้งาน และทำการทดสอบการทำงานของระบบ

ผู้รับผิดชอบ : นายอภิรัตน์ พุทธิรักษา

## 7.พัฒนาและทดสอบระบบการคำนวณดอกเบี้ย

ออกแบบและพัฒนาการคำนวณอัตราดอกเบี้ย ส่วนติดต่อผู้ใช้งาน และทำการทดสอบการทำงานของระบบ

ผู้รับผิดชอบ : นายพีรวิชญ์ ประพันธ์วิทยา

## 8.ทดสอบการทำงานของระบบโดยรวม

ทดสอบการทำงานและตรวจสอบความถูกต้องของระบบทั้งหมด

ผู้รับผิดชอบ : นายพีรวิชญ์ ประพันธ์วิทยา และ นายอภิรัตน์ พุทธรักษา

## 9.จัดทำเอกสารและส่งชิ้นงาน

จัดทำเอกสารประกอบชิ้นงาน และจัดส่งชิ้นงาน

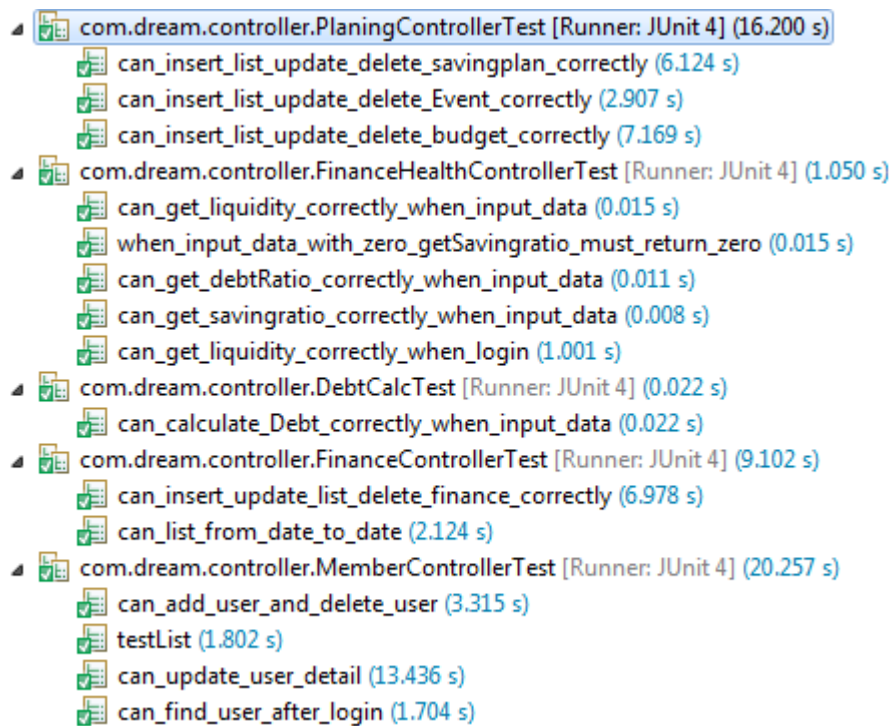
ผู้รับผิดชอบ : นายพีรวิชญ์ ประพันธ์วิทยา และ นายอภิรัตน์ พุทธรักษา

## ผลการทดสอบซอฟต์แวร์

### Unit Test

Test Directory : src/main/test

#### ส่วน Back End



ภาพที่ 3 ผลการทดสอบ Unit test

#### 1. ระบบสมาชิก

การทดสอบ	ผลการทดสอบ
สามารถเพิ่มข้อมูลสมาชิกได้ถูกต้อง	สามารถทำงานได้ถูกต้อง
สามารถแก้ไขข้อมูลสมาชิกได้ถูกต้อง	สามารถทำงานได้ถูกต้อง
สามารถลบข้อมูลสมาชิกได้ถูกต้อง	สามารถทำงานได้ถูกต้อง
สามารถค้นหาข้อมูลสมาชิกได้ถูกต้อง	สามารถทำงานได้ถูกต้อง

ตารางที่ 2 ผลการทดสอบระบบสมาชิก



#### 4. คำนวนดอกเบี้ย

การทดสอบ	ผลการทดสอบ
สามารถคำนวณดอกเบี้ยเงินฝากและดอกเบี้ยเงินกู้ได้ถูกต้องตามข้อมูลที่ระบุ	สามารถทำงานได้ถูกต้อง

ตารางที่ 5 ผลการทดสอบการคำนวณดอกเบี้ย

#### 5. คำนวนสถานภาพทางการเงิน

การทดสอบ	ผลการทดสอบ
สามารถดึงข้อมูลผู้ใช้งานจากฐานข้อมูลและนำมาวิเคราะห์ได้อย่างถูกต้อง	สามารถทำงานได้ถูกต้อง
สามารถรับข้อมูลจากผู้ใช้งานและนำมาวิเคราะห์ได้อย่างถูกต้อง	สามารถทำงานได้ถูกต้อง

ตารางที่ 6 ผลการทดสอบการคำนวณสถานภาพทางการเงิน

### Evaluation

#### 1. ทดสอบการเพิ่มรายจ่ายผ่านทางหน้าเว็บไซต์

**จุดประสงค์ของการทดลอง :** เพื่อทำการทดสอบว่าข้อมูลรายจ่ายบันทึกลงในฐานข้อมูล สามารถนำมาแสดงผลได้อย่างถูกต้อง และถ้ามีการบันทึกงบประมาณของรายรับรายจ่ายประเภทนั้นไว้จะสามารถคำนวณจำนวนเงินคงเหลือภายในงบประมาณได้ถูกต้อง

**สิ่งที่จะวัด :** ความถูกต้องของการบันทึกรายรับรายจ่าย เช่น จำนวนเงิน วันที่ทำการบันทึก รายละเอียดของการบันทึก เป็นต้น และจำนวนเงินคงเหลือในงบประมาณที่ได้ทำการบันทึกไว้

**วิธีการและสิ่งที่ใช้ในการทดลอง :**

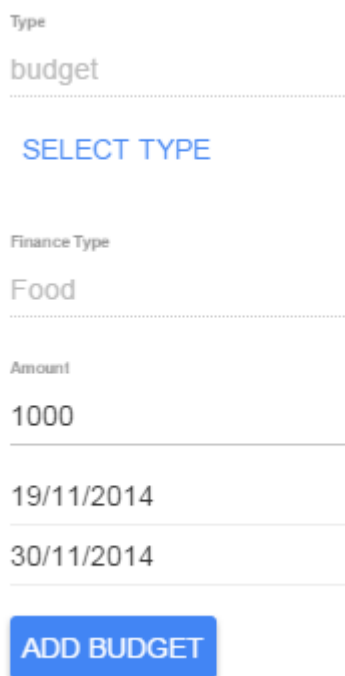
- 1.ทำการบันทึกงบประมาณของรายจ่ายที่ต้องการ
- 2.ทำการบันทึกรายจ่ายให้แก่ระบบผ่านทางหน้าเว็บไซต์
3. ตรวจสอบข้อมูลรายรับรายจ่ายที่สามารถเพิ่มลงได้ฐานข้อมูล แสดงผลได้ถูกต้อง และจำนวนเงินที่เหลือในงบประมาณสามารถคำนวณได้ถูกต้อง
4. ออกจากระบบและเข้าระบบใหม่อีกครั้งจากนั้นทำการตรวจสอบข้อมูลในข้อที่ 3

5. บันทึกผลการทดสอบ

6. ทดสอบการบันทึกรายจ่ายอีกครั้งตามที่ต้องการ

ผลที่ได้จากการทดลอง :

สามารถบันทึกรายจ่ายและทำการคำนวณงบประมาณคงเหลือได้อย่างถูกต้อง และเมื่อทำการเข้าสู่ระบบอีกครั้งข้อมูลยังคงถูกต้อง



Type  
budget

SELECT TYPE

Finance Type  
Food

Amount  
1000

19/11/2014

30/11/2014

ADD BUDGET

ภาพที่ 4 ภาพการบันทึกข้อมูลงบประมาณ

Start Date	End Date	For	Use	Limit	Budget Left	
2014-11-19	2014-11-30	Food	0	1000	+1000	

EDIT

ภาพที่ 5 ภาพผลการบันทึกงบประมาณ

Date	Amount	Type	Description
2014/11/19	1500	Salary	
2014/11/19	100	Food	
2014/11/19	1000	Travel	

EDIT

ภาพที่ 6 ภาพข้อมูลรายรับรายจ่ายก่อนการบันทึกรายจ่าย

Type  
outcome

SELECT TYPE

Finance Type  
Food

Amount  
100

WITH EVENT

Detail  
suki

ADD

ภาพที่ 7 ภาพการบันทึกข้อมูลงบประมาณ



Date	Amount	Type	Description
2014/11/19	1500	Salary	
2014/11/19	100	Food	
2014/11/19	1000	Travel	
2014/11/19	100	Food	suki

EDIT

ภาพที่ 8 ภาพผลการบันทึกรายจ่าย

Start Date	End Date	For	Use	Limit	Budget Left
2014-11-19	2014-11-30	Food	100	1000	+900

EDIT

ภาพที่ 9 ผลการคำนวณงบประมาณคงเหลือเมื่อทำการบันทึกรายจ่าย

Date	Amount	Type	Description
2014/11/19	1500	Salary	
2014/11/19	100	Food	
2014/11/19	1000	Travel	
2014/11/19	100	Food	suki

EDIT

ภาพที่ 10 ผลการบันทึกรายจ่ายหลังจากทำการเข้าระบบอีกครั้ง

Start Date	End Date	For	Use	Limit	Budget Left	
2014-11-19	2014-11-30	Food	100	1000	+900	
						

ภาพที่ 11 ผลการคำนวณงบประมาณคงเหลือเมื่อทำการเข้าระบบอีกครั้ง

**สรุปผลการทดลอง :** จากการทดลองพบว่าสามารถเพิ่มข้อมูลรายจ่ายและคำนวณจำนวนเงินคงเหลือของงบประมาณได้ถูกต้อง

## 2. ทดสอบ Response Time ของ Back End แต่ละส่วนและความคงทนของระบบ

**จุดประสงค์ของการทดลอง :** เพื่อตรวจสอบเวลาในการตอบสนอง (Response Time) โดยเฉลี่ยในแต่ละส่วนของระบบ และเมื่อมี ร้องขอ (Request) เป็นจำนวนมากระบบจะสามารถตอบสนองการทำงานได้อย่างรวดเร็วและถูกต้อง

**สิ่งที่จะวัด :** เวลาในการตอบสนอง (Response Time) โดยเฉลี่ยและการรองรับจำนวนผู้ใช้งานเป็นจำนวนมาก

**วิธีการและสิ่งที่ต้องใช้ในการทดลอง :** ใช้ Apache Jmeter เพื่อทำการร้องขอ (Request) เป็นจำนวนมากไปยังส่วน Restful web service และทำการเก็บระยะเวลาในการตอบสนอง (Response Time) ของระบบ

ผลที่ได้จากการทดลอง :

ผลการทดลองจาก Apache JMeter

การเชื่อมต่อที่ใช้	:	My By Cat 384 Kbps
Thread ที่ใช้ในการเชื่อมต่อ	:	50
Loop/Thread	:	1000
Request/Loop	:	22
เครื่องที่ใช้ในการทดสอบ	:	Intel Core i7-2630 QM Quad-core with hyper threading Memory 8 GB DDR3

Label	# Samples	Average	Min	Max	Error %
REST_API:j_spring_security_check	939	2308	15	326281	4.37%
REST_API:/member/update	934	3456	74	320083	5.35%
REST_API:/finance/insert	933	1839	21	322123	5.79%
REST_API:/finance/list	930	2432	83	351134	6.67%
REST_API:/finance/update	926	2305	0	320817	7.24%
REST_API:/finance/delete	924	1932	85	348893	7.25%
REST_API:/planing/saving/insert	922	646	35	55616	7.05%
REST_API:/planing/saving/list	921	993	82	315102	6.62%
REST_API:/planing/saving/edit/	920	1740	81	316428	8.15%
REST_API:/planing/saving/delete/	918	2869	65	698973	8.93%
REST_API:/planing/event/insert	914	682	80	59416	7.88%
REST_API:/planing/event/list	912	1585	70	343912	8.11%
REST_API:/planing/event/edit/	909	1237	76	318155	9.13%
REST_API:/planing/event/delete/	907	1543	90	315282	9.04%
REST_API:/planing/budget/insert	905	1655	85	290137	9.28%
REST_API:/planing/budget/list	902	1241	35	319749	9.09%
REST_API:/planing/budget/edit/	902	1502	92	366070	9.87%
REST_API:/planing/budget/delete/	897	2170	84	347179	10.26%
REST_API:j_spring_security_logout	895	1367	62	340567	1.79%
REST_API:/financehealth/getliquidity	891	917	74	322087	1.35%
REST_API:/financehealth/debtratio	890	2344	73	318306	1.57%
REST_API:/financehealth/savingratio	890	1672	71	320911	1.80%
TOTAL	20081	1751	0	698973	6.67%

ตารางที่ 12 ผลการทดสอบประสิทธิภาพการทำงานของ Web service

Text

- REST\_API:/planing/event/edit/
- REST\_API:/spring\_security\_check
- REST\_API:/finance/list
- REST\_API:/financehealth/getliquidity
- REST\_API:/planing/budget/insert
- REST\_API:/finance/update
- REST\_API:/planing/event/list
- REST\_API:/planing/budget/insert
- REST\_API:/financehealth/savingratio
- REST\_API:/planing/budget/edit/
- REST\_API:/planing/budget/insert
- REST\_API:/planing/budget/edit/
- REST\_API:/planing/budget/edit/
- REST\_API:/spring\_security\_logout
- REST\_API:/finance/delete
- REST\_API:/spring\_security\_check
- REST\_API:/planing/budget/delete/
- REST\_API:/spring\_security\_check
- REST\_API:/member/update
- REST\_API:/planing/saving/delete/
- REST\_API:/planing/event/delete/
- REST\_API:/spring\_security\_logout
- REST\_API:/planing/saving/edit/
- REST\_API:/planing/saving/delete/
- REST\_API:/planing/event/delete/
- REST\_API:/planing/event/insert
- REST\_API:/planing/event/list
- REST\_API:/finance/insert
- REST\_API:/planing/saving/edit/
- REST\_API:/spring\_security\_check
- REST\_API:/planing/event/insert
- REST\_API:/finance/update
- REST\_API:/spring\_security\_logout
- REST\_API:/finance/list
- REST\_API:/spring\_security\_check
- REST\_API:/planing/budget/edit/
- REST\_API:/planing/saving/insert
- REST\_API:/finance/list
- REST\_API:/planing/budget/edit/
- REST\_API:/planing/saving/list
- REST\_API:/planing/event/list
- REST\_API:/planing/budget/delete/
- REST\_API:/finance/delete
- REST\_API:/finance/insert
- REST\_API:/spring\_security\_logout
- REST\_API:/planing/saving/delete/
- REST\_API:/planing/saving/delete/

☐ Scroll automatically?

Sampler result   Request   Response data

Thread Name: Thread Group 1-49  
Sample Start: 2557-11-19 04:43:01 ICT  
Load time: 349570  
Latency: 9728  
Size in bytes: 1345  
Headers size in bytes: 0  
Body size in bytes: 0  
Sample Count: 1  
Error Count: 1  
Response code: Non HTTP response code: java.net.SocketException  
Response message: Non HTTP response message: Socket closed

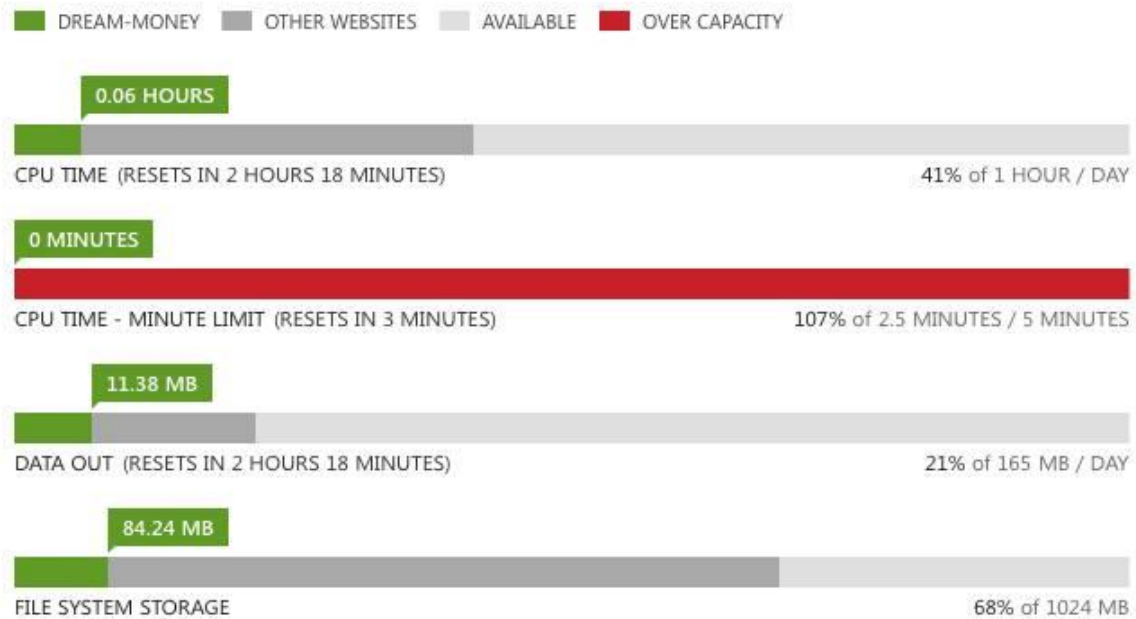
Response headers:

SampleResult fields:  
ContentType:  
DataEncoding: null

Raw   Parsed

ภาพที่ 13 ผลลัพธ์ของการทำงานเมื่อระบบไม่สามารถตอบสนองการใช้งานได้เมื่อมีการร้องขอเป็นจำนวนมาก

## ผลการทำงานของ Azure cloud websites



ภาพที่ 14 สถานะของระบบ Website

\*หมายเหตุ จำนวน Sample ที่ได้นั้นไม่ตรงตามที่ได้ระบุจำนวนการทำงานไว้เนื่องจาก website นั้นหยุดการทำงานจึงทำให้จำเป็นต้องหยุดการทดสอบไว้เพียง 20081 Sample เท่านั้น

**สรุปผลการทดลอง :** จากการทดลองค่าเฉลี่ยระยะเวลาการตอบสนองของระบบมีค่าเป็น 1751 ms ระยะเวลาการตอบสนองสูงสุดของระบบมีค่าเป็น 698973 ms และอัตราความผิดพลาดในการทำงาน 6.67% เนื่องจากเมื่อมีการขอการทำงาน (Request) เป็นจำนวนมากซึ่งทำให้ระบบไม่สามารถรองรับการทำงานได้โดยสามารถสังเกตได้จากภาพที่ 3 ซึ่ง Apache JMeter ระบุว่าไม่สามารถเชื่อมต่อไปยัง web service ได้ และในภาพที่ 4 CPU ของส่วน Web service มีการใช้งานถึง 107% ทำให้ระยะเวลาในการตอบสนองนานขึ้นเนื่องจากไม่สามารถประมวลผลการทำงานได้ทัน อีกทั้งยังทำให้เกิดข้อผิดพลาดในการทำงานถึง 6.67% แต่ถ้าทำการอัปเกรดการใช้บริการ Azure: Microsoft's Cloud Platform แบบ Web Sites ให้รองรับการประมวลผลที่มากขึ้นจะช่วยให้สามารถแก้ไขปัญหาที่เกิดขึ้นได้

## บทสรุป

Money Logging เป็นเว็บแอปพลิเคชันที่ช่วยในการบันทึกรายรับรายจ่ายให้แก่ผู้ใช้งานเพื่อให้สามารถตรวจสอบการใช้เงินของตนเองได้ โดยภายในเว็บแอปพลิเคชันนั้นมีเครื่องมือที่ช่วยในการคำนวณอัตราดอกเบี้ย อัตราส่วนดอกเบี้ยต่อเงินออมทั้งหมด อัตราส่วนเงินออมจากรายได้ทั้งหมดต่อปี การวางแผนทางการเงินต่างๆ อีกทั้งยังสามารถประเมินสถานภาพทางการเงินของผู้ใช้งานโดยทำการตรวจสอบจากข้อมูลรายรับรายจ่ายหรือให้ผู้ใช้งานป้อนข้อมูลให้กับระบบเองได้ ซึ่งจะช่วยให้ผู้ใช้งานนำข้อมูลดังกล่าวไปใช้ในการตัดสินใจในการวางแผนทางการเงินในอนาคตได้

Money Logging ระบบภายในนั้นจะการทำงานแยกออกเป็น 2 ส่วนคือ ส่วน Presentation คือส่วนที่ใช้ในการติดต่อกับผู้ใช้งาน และส่วนที่ 2 คือส่วน Business logic ซึ่งเป็นส่วนที่ใช้ในการประมวลผลต่างๆที่ได้กล่าวไปในการออกแบบระบบ (System design) โดยจะใช้ RESTfull API ในการเชื่อมต่อการทำงานระหว่าง 2 ส่วน ซึ่งทำให้สามารถใช้ระบบใดก็ได้ที่รองรับการเรียกใช้งาน RESTfull API มาติดต่อกับ Business Logic และยังสามารถขยายตัวในอนาคตเพื่อรองรับการใช้งานจากผู้ใช้งานเป็นจำนวนมากได้

ในการพัฒนาเว็บแอปพลิเคชัน Money Logging นั้นให้ผู้พัฒนาได้รับความรู้ความเข้าใจในการใช้งานระบบ cloud computing และกระบวนการพัฒนาซอฟต์แวร์เพื่อนำไปใช้งานจริงมากยิ่งขึ้นและสามารถนำประสบการณ์ไปปรับใช้ในการทำงานในอนาคตได้

## บรรณานุกรม

- [1] Google Inc.2010-2014. AngularJS. Available at: <https://angularjs.org/> (20 กันยายน 2557)
- [2] Polymer Authors 2014. Polymer. Available at: <https://www.polymer-project.org/> (10 กันยายน 2557)
- [3] Pivotal Software, Inc. Spring Framework Reference Documentation. Available at: <http://docs.spring.io/spring/docs/4.1.3.BUILD-SNAPSHOT/spring-framework-reference/htmlsingle/> (15 ตุลาคม 2557)
- [4] Pivotal Software, Inc. Spring Security Reference. Available at: <http://docs.spring.io/spring-security/site/docs/4.0.0.M2/reference/htmlsingle/> (30 ตุลาคม 2557)
- [5] Apache Software Foundation. Apache JMeter. Available at: <http://jmeter.apache.org/> (10 พฤศจิกายน 2557)
- [6] Microsoft Corporation. Azure. Available at: <http://azure.microsoft.com/en-us/> (10 กันยายน 2557)
- [7] Angular Material. Available at: <https://material.angularjs.org/#/> (20 ตุลาคม 2557)
- [8] Bootstrap. Available at: <http://getbootstrap.com/> (5 ตุลาคม 2557)
- [9] The Apache Software Foundation. Maven. Available at: <http://maven.apache.org/> (18 ตุลาคม 2557)
- [10] Bower. Available at: <http://bower.io/> (2 พฤศจิกายน 2557)