Project

เรื่อง ระบบบริหารจัดการห้องสมุดออนไลน์

กลุ่ม ParatabPlus

สมาชิกในกลุ่ม

1. นายสุรวิทย์ แสงจันทร์ รหัสนักศึกษา 55011357

2. นายสุรัชนันท์ ไกรเดช รหัสนักศึกษา 55011362

โครงงานนี้เป็นส่วนหนึ่งของรายวิชา

01076254 – การวิเคราะห์และออกแบบเชิงวัตถุ

(Object-oriented Analysis and Design)

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ภาคเรียนที่ 1 ปีการศึกษา 2557

Object-oriented Analysis and Design Fall 1/2014 @ KMITL

Team name ParatabPlus

Project Online library book system (ระบบบริหารจัดการห้องสมุดออนไลน์)

Group members

1. นายสุรวิทย์ แสงจันทร์ รหัสนักศึกษา 55011357
2. นายสุรัชนันท์ ไกรเดช รหัสนักศึกษา 55011362

GitHub repository name Lib_booksys

Abstract

ระบบบริหารจัดการห้องสมุดออนไลน์นี้มีจุดประสงค์คือต้องการช่วยเหลือระบบงานห้องสมุดทั้งฝั่งบรรณารักษ์และสมาชิกที่ใช้งานให้เกิดความสะดวกสบาย แก้ปัญหาต่างๆ ที่ระบบงานห้องสมุดเกิดขึ้น เช่น การค้นหาและดูข้อมูลหนังสือ การติดตามการยืมและคืนหนังสือ โดยระบบงานห้องสมุดที่พัฒนานั้นจะประกอบไปด้วย ระบบจัดการข้อมูลหนังสือ (การเพิ่ม,แก้ไข,ลบและค้นหา) ที่มีอยู่ในห้องสมุด ระบบการจองหนังสือที่ต้องการ ระบบต่ออายุการยืมหนังสือ ระบบการยืม/การคืน ระบบแจ้งเตือนหนังสือที่ยืมเกินกำหนด พร้อมกับส่วนติดต่อผู้ใช้งานที่เรียบง่าย

Introduction and motivation

ห้องสมุดคือสถานที่ที่มีผู้คนต่างเข้ามาใช้บริการเพื่อสืบค้นหาความรู้ในด้านต่างๆ ผู้ใช้บริการในห้องสมุดมีหลายรูปแบบ มีทั้งเข้ามาหาหนังสืออ่านปกติ และหาหนังสือที่ต้องการอ่านเพื่อยืมกลับไปอ่านที่อื่น บ่อยครั้งที่ผู้ใช้งานและบรรณารักษ์เกิดสถานการณ์ที่ไม่เอื้ออำนวยเช่น ไม่เจอหนังสือที่ต้องการอ่าน(เพราะหนังสือที่หาไม่มีอยู่จริงในห้องสมุด) ต้องการยืมหนังสือที่หมายตาไว้แต่มีผู้ใช้บริการคนอื่นมายืมไป หรือแม้แต่ผู้ใช้บริการยืมหนังสือไปแล้วไม่นำกลับมาคืนก็ดี สถานการณ์ต่างๆที่กล่าวมาทำให้ผู้ใช้บริการเสียเวลา เพราะผู้ใช้บริการแต่ละคนต่างมีเวลาว่างไม่เหมือนกัน ส่วนผู้บริหารจัดการก็ต้องเสียเวลาในการติดตามหนังสือและส่งผลกระทบต่อผู้ใช้บริการที่ต้องการอ่านหรือยืมหนังสือที่ถูกยืมไปอีกด้วย โครงงาน Web application นี้จึงมีจุดประสงค์ที่ต้องการพัฒนาระบบบริหารจัดการห้องสมุดในด้านต่างๆ เพื่อช่วยเหลือปัญหาในสถานการณ์ต่างๆที่กล่าวมา อีกทั้งผู้จัดทำต้องการศึกษาวิธีการทำระบบห้องสมุดนี้และต้องการศึกษาหาความรู้วิธีการพัฒนา Web application โดยใช้ Framework และภาษาโปรแกรมที่ต้องการในการพัฒนา การทำงานหลักๆของระบบจึงมุ่งเน้นไปที่หัวใจของระบบงานห้องสมุดเช่น ระบบฐานข้อมูลหนังสือ การค้นหา/เพิ่ม/แก้ไข/ลบ รายชื่อหนังสือที่มีในห้องสมุด ระบบสมาชิก ระบบการจองหนังสือสำหรับผู้ที่ไม่สะดวกมายืมทันทีแต่ต้องการจะจองเพื่อที่จะยืมในภายหลัง ระบบการแจ้งเตือนหนังสือที่ยืมไป(โดยเฉพาะการยืมหนังสือที่เกินกำหนด) นอกจากความสามารถในระบบหลักที่จะทำแล้ว ระบบที่คิดว่าจะทำเป็นส่วนขยายเพิ่มเติมก็อาจจะเป็นสถิติผู้ใช้บริการที่ยืมหนังสือมากที่สุด สถิติการยืมหนังสือยอดนิยม เป็นต้น และอีกสิ่งหนึ่งที่หลีกเลี่ยงไม่ได้ในการพัฒนา Web application ไม่ว่าจะเป็นระบบใดๆก็ตามก็คือส่วนติดต่อผู้ใช้งานซึ่งเป็นอีกส่วนหนึ่งที่ต้องการออกแบบเพื่อตอบโจทย์ความสะดวกสบายแก่ผู้ที่เข้ามาใช้งาน

Related work

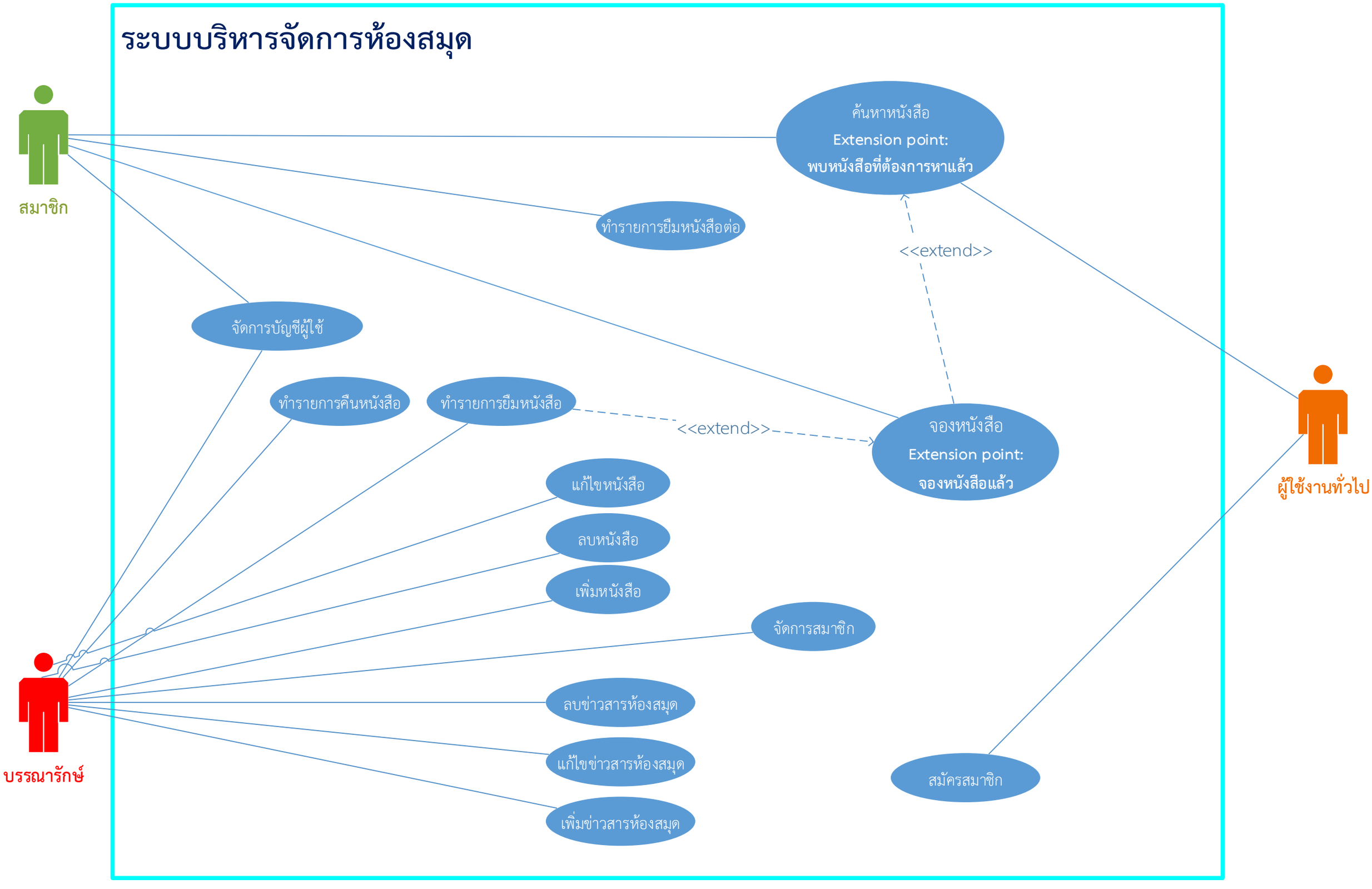
สำหรับงานที่เกี่ยวข้องกับงานที่จะทำใน Project ตัวอย่างเช่น เว็บไซต์ห้องสมุดโรงเรียนสวนกุหลาบวิทยาลัย (library.sk.ac.th) หอสมุดแห่งชาติ (www.nlt.go.th) สำนักหอสมุดกลาง สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง (www.lib.kmitl.ac.th) สำนักงานวิทยทรัพยากร จุฬาลงกรณ์มหาวิทยาลัย (www.car.chula.ac.th) และ UCLA Library (www.library.ucla.edu) ความคล้ายกันของเว็บไซต์เหล่านี้กับงานที่จะทำประกอบไปด้วย ระบบสืบค้นหนังสือภายในห้องสมุด ที่จะแสดงผลการค้นหาของสถานะของหนังสือที่ผู้ใช้บริการต้องการ ระบบจัดการผู้ใช้ ซึ่งจะสามารถทำให้ผู้ใช้บริการสามารถตรวจสอบสถานะของหนังสือที่ยืมมาได้ วันครบกำหนดการคืนหนังสือ หรือสถานะของการจองหนังสือเล่มที่ผู้ใช้ห้องสมุดต้องการ รวมไปถึงสามารถทำรายการยืมหนังสือเล่มนั้นต่อออกไปได้อีกโดยที่ไม่จำเป็นต้องเดินทางไปทำรายการยังห้องสมุด หน้าเว็บซึ่งจะใช้ลงประกาศต่างๆจากทางห้องสมุดให้ผู้ใช้บริการห้องสมุดสามารถรับทราบ เช่น ประกาศรายชื่อหนังสือใหม่ในแต่ละเดือน กิจกรรมของทางห้องสมุดที่จะจัดขึ้น กฎการใช้ห้องสมุดหรือข้อมูลต่างๆที่ผู้ใช้บริการห้องสมุดควรรู้ เป็นต้น สิ่งที่แตกต่างกันจากงานที่เกี่ยวข้องคือรูปแบบในการค้นหาหนังสือ(ค้นหาโดยใช้ผู้แต่ง/ชื่อเรื่อง/ปีที่พิมพ์/อื่นๆ) การออกแบบส่วนติดต่อผู้ใช้ รูปแบบของการพัฒนาและส่วนขยายเพิ่มเติมที่ได้กล่าวไปในบทนำ เช่น จัดอันดับผู้ที่ยืมหนังสือมากที่สุดและจัดอันดับหนังสือที่ถูกยืมมากที่สุด

ผลการวิเคราะห์ความต้องการของผู้ใช้ระบบ

- **Functional requirement**
    1. ผู้ใช้งานสามารถค้นหาหนังสือและดูรายละเอียดของหนังสือที่มีอยู่ในห้องสมุดได้
    2. สมาชิกสามารถต่ออายุการยืมของหนังสือได้ไม่เกิน 3 ครั้ง
    3. สมาชิกสามารถจองหนังสือที่สมาชิกคนอื่นกำลังยืมอยู่ได้
    4. สมาชิก/บรรณารักษ์สามารถแก้ไขข้อมูลส่วนตัวได้
    5. บรรณารักษ์สามารถเพิ่ม/แก้ไข/ลบรายการหนังสือที่มีอยู่ในห้องสมุดได้
    6. บรรณารักษ์สามารถดูรายชื่อและลบสมาชิกในห้องสมุดได้
    7. บรรณารักษ์สามารถเพิ่ม/ดูรายชื่อและลบรายชื่อบรรณารักษ์ในห้องสมุดได้
    8. บรรณารักษ์สามารถเพิ่ม/แก้ไข/ลบข่าวสารของห้องสมุดได้
- **Non-functional requirement**
    1. มีการใช้งานทุกๆส่วนของระบบห้องสมุดที่สะดวกสบาย
    2. มีส่วนติดต่อผู้ใช้งานที่เรียบง่าย
    3. สามารถรองรับการใช้งานจากผู้ใช้ได้หลายคนพร้อมกัน
    4. มีการรักษาความปลอดภัยของระบบและข้อมูลผู้ใช้งานที่ดี
    5. สามารถเรียกค้นหาข้อมูลและเรียกดูข้อมูลรวมไปถึงจัดการข้อมูลที่เกี่ยวกับหนังสือได้ในเวลารวดเร็วและแม่นยำ

แนวทางการใช้งาน

- Use case diagram



- **Use case specification**

[Use case 1]

ชื่อ Use case

ค้นหาหนังสือ(Search book)

วัตถุประสงค์ของ Use case

Use case นี้มีวัตถุประสงค์เพื่อให้ระบบสามารถให้ผู้ใช้ค้นหาหนังสือโดยค้นหาจากคำสำคัญ (keyword) ที่ต้องการได้

เงื่อนไขที่เป็นจริงก่อนทำ Use case

ระบบออนไลน์อยู่และผู้ใช้มีคำสำคัญที่ต้องการค้นหา

เงื่อนไขที่เป็นจริงหลังทำ Use case

รายชื่อหนังสือต่างๆที่พบจากคำสำคัญที่ผู้ใช้ให้มา

ข้อจำกัด

ไม่มี

ขั้นตอนการทำงานใน Success scenario

A. ผู้ใช้เข้ามาที่หน้าเว็บเลือกค้นหา
B. ผู้ใช้พิมพ์คำสำคัญที่ต้องการค้นหาหนังสือลงในช่องค้นหา
C. ผู้ใช้กดส่งคำสำคัญที่ต้องการค้นหา
D. ระบบทำการค้นหาหนังสือตามคำสำคัญที่ให้มา
E. ระบบส่งผลลัพธ์การค้นหากลับไปให้ผู้ใช้
F. ผู้ใช้ดูผลลัพธ์การค้นหาหนังสือ
G. จบการทำงาน

ขั้นตอนการทำงานใน Alternative scenario

เงื่อนไขที่ทำให้เกิด Alternative scenario

เงื่อนไขที่ 1:ไม่พบผลลัพธ์การค้นหา

E1. ระบบแจ้งให้ผู้ใช้ทราบว่าไม่พบผลลัพธ์ที่ต้องการค้นหา
E2. ผู้ใช้รับทราบผลลัพธ์ที่เกิดขึ้น
E3. ผู้ใช้ตัดสินใจกับสิ่งที่เกิดขึ้น
E4.ถ้าผู้ใช้ต้องการลองใหม่ค้นหาอีกครั้ง กลับไปที่ Success scenario ขั้นตอน B

E5.ถ้าผู้ใช้ไม่ต้องการลองใหม่ ดำเนินการขั้นตอนต่อไป

E6. จบการทำงาน

เงื่อนไขที่ 2:ผู้ใช้ต้องการดูรายละเอียดหนังสือที่ต้องการ

F1. ผู้ใช้กดเลือกดูหนังสือที่ต้องการ

F2. ระบบเรียกข้อมูลหนังสือที่ผู้ใช้ต้องการดู

F3. ระบบแสดงรายละเอียดข้อมูลหนังสือให้ผู้ใช้ทราบ

F4. ผู้ใช้ดูรายละเอียดหนังสือ

F5. จบการทำงาน

จุดขยาย(Extension point) – จองหนังสือ(Reserve book)

เงื่อนไขที่ 3:ผู้ใช้ต้องการจองหนังสือ

F5.1. ผู้ใช้ดูสถานะของหนังสือ

F5.2. ถ้าหนังสือมีคนยืมอยู่แต่ไม่มีคนจอง ไปที่ขั้นตอน F5.4

F5.3. ถ้าหนังสือไม่มีคนยืมหรือมีคนจองแล้ว ไปที่ขั้นตอน F5.8

F5.4. ผู้ใช้ดำเนินการจองหนังสือ

F5.5. ระบบทำการบันทึกข้อมูลการจองหนังสือ

F5.6. ระบบแจ้งให้ผู้ใช้ทราบว่าได้ดำเนินการจองเรียบร้อยแล้ว ให้ผู้ใช้มาติดต่อยืมที่ห้องสมุดภายในเวลากำหนด

F5.7. ผู้ใช้รับทราบผลการจอง

F5.8. จบการทำงาน

<u>สมมติฐาน</u>

การค้นหาหนังสือจะทำได้ก็ต่อเมื่อผู้ใช้มีคำค้นหาที่ต้องการค้น ผลลัพธ์การค้นหาก็จะสอดคล้องกับข้อมูลหนังสือที่มีอยู่ในห้องสมุด

[Use case 2]

ชื่อ Use case

จองหนังสือ (Reserve book)

วัตถุประสงค์ของ Use case

Use case นี้มีวัตถุประสงค์เพื่อให้ระบบสามารถให้ผู้ใช้จองหนังสือที่ต้องการ แต่หนังสือเล่มดังกล่าวอยู่ในระหว่างการยืมของผู้ใช้ห้องสมุดรายอื่น

เงื่อนไขที่เป็นจริงก่อนทำ Use case

ระบบออนไลน์อยู่และหนังสือที่ต้องการอยู่ในระหว่างการยืมของผู้อื่น

เงื่อนไขที่เป็นจริงหลังทำ Use case

หนังสือถูกจองไว้ให้กับผู้ที่ทำการจอง

ข้อจำกัด

การจองหนังสือได้ จะต้องมีหนังสืออยู่ในห้องสมุดและหนังสือเล่มนั้นต้องอยู่ในสถานะที่ถูกยืม

ขั้นตอนการทำงานใน Success scenario

H. ผู้ใช้เข้ามาที่หน้าเว็บเลือกค้นหา
I. ผู้ใช้พิมพ์คำสำคัญที่ต้องการค้นหาหนังสือลงในช่องค้นหา
J. ผู้ใช้กดส่งคำสำคัญที่ต้องการค้นหา
K. ระบบทำการค้นหาหนังสือตามคำสำคัญที่ให้มา
L. ระบบส่งผลลัพธ์การค้นหากลับไปให้ผู้ใช้
M. ผู้ใช้เลือกหนังสือที่ต้องการจากผลลัพธ์ที่แสดง
N. ผู้ใช้ตรวจดูสถานะของหนังสือ พบว่าอยู่ในระหว่างการยืม
O. ผู้ใช้กดจองหนังสือเล่มดังกล่าว
P. ระบบให้ผู้ใช้กรอกข้อมูลที่จำเป็น ผู้ใช้กรอกข้อมูล และกดส่งข้อมูล
Q. ระบบทำการจองหนังสือเล่มดังกล่าวสำหรับผู้ใช้ที่ร้องขอ

ขั้นตอนการทำงานใน Alternative scenario

เงื่อนไขที่ทำให้เกิด Alternative scenario

เงื่อนไขที่ 1:หนังสือไม่ได้อยู่ในสถานะถูกยืม และหนังสืออยู่ที่ชั้นวางหนังสือ

G1.1 ระบบแจ้งให้ผู้ใช้ทราบว่าหนังสือเล่มที่ต้องการอยู่ที่ชั้นวางหนังสือ

G1.2 ผู้ใช้รับทราบสถานะของหนังสือ

G1.3 ผู้ใช้ตัดสินใจไปหยิบหนังสือที่ชั้นหนังสือ

G1.4 จบการทำงาน

เงื่อนไขที่ 2:หนังสือไม่ได้อยู่ในสถานะถูกยืม และหนังสือไม่อยู่ในห้องสมุด (ในกรณีอยู่ในระหว่างการติดตาม หรืออยู่ในระหว่างซ่อมแซม)

G2.1 ระบบแจ้งให้ผู้ใช้ทราบสถานะของหนังสือ

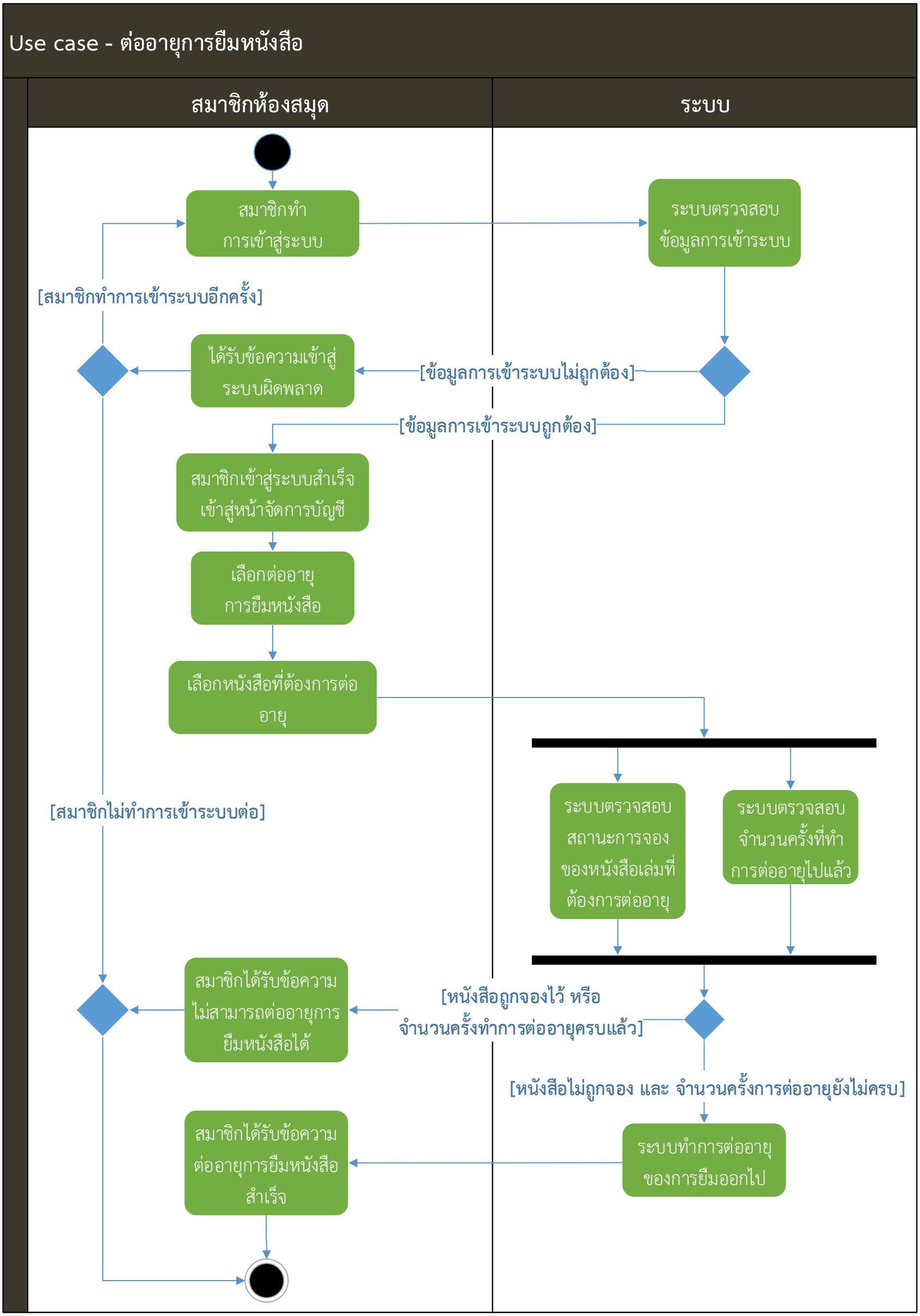
G2.2 ผู้ใช้รับทราบสถานะของหนังสือ

G2.3 จบการทำงาน

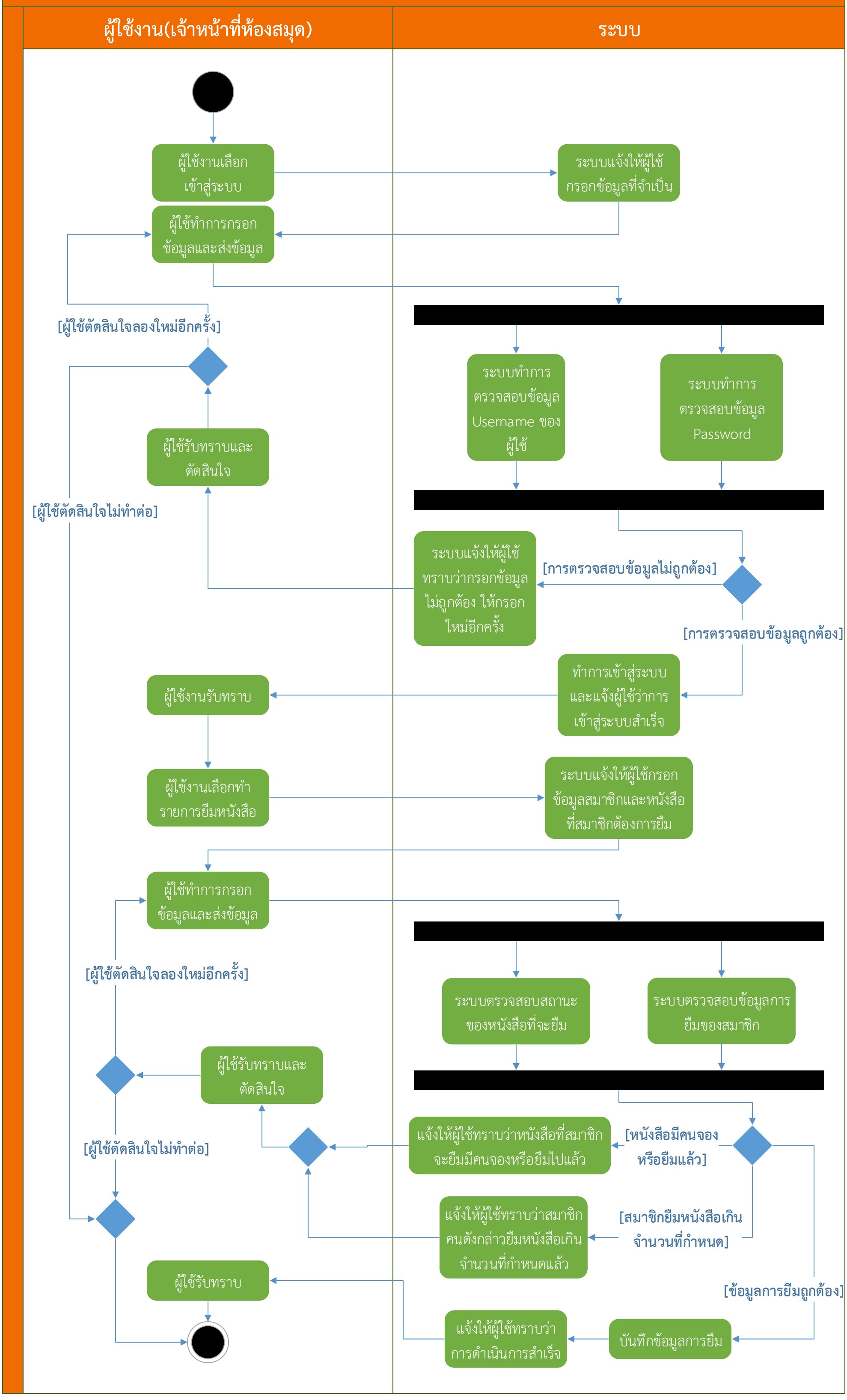
<u>สมมติฐาน</u>

ผู้ใช้(สมาชิก)สามารถจองหนังสือเล่มที่ต้องการได้ ถ้าหนังสือเล่มนั้นอยู่ในระหว่างการยืมโดยผู้ใช้คนอื่น หลังจากการจองสำเร็จแล้วผู้ใช้ก็จะมีสิทธิ์ยืมหนังสือเล่มนั้นต่อได้ทันทีที่ผู้ใช้คนก่อนหน้าคืนหนังสือที่จอง

- Activity diagram



Use case - ทำรายการยืมหนังสือ



สถาปัตยกรรมของระบบ

- Problem Analysis

จากการวิเคราะห์ระบบในภาพรวมแล้ว สามารถกำหนด abstraction ของแต่ละส่วนของระบบได้เป็นดังนี้

1. หนังสือ ประกอบไปด้วยรหัสหนังสือ หมายเลขเรียกหนังสือ ชื่อหนังสือ ชื่อผู้แต่ง สำนักพิมพ์ รายละเอียด ปีที่พิมพ์ และสถานะของหนังสือ
2. ข่าวสาร ประกอบไปด้วยรหัสข่าว หัวข้อข่าว เนื้อหาข่าว เวลาที่ประกาศข่าว
3. รายการยืม-คืนหนังสือ เป็นรายการเก็บข้อมูลการยืมและคืนของผู้ใช้งานในห้องสมุด ประกอบไปด้วยเลขลำดับการยืม รหัสผู้ใช้งานที่ยืม รหัสหนังสือที่ยืม วันที่ยืม กำหนดคืน วันที่คืน และจำนวนครั้งในการต่ออายุการยืม
4. รายการจองหนังสือ เป็นรายการที่เก็บข้อมูลการจองหนังสือของผู้ใช้งาน ประกอบไปด้วย รหัสผู้ใช้งานที่จอง รหัสหนังสือที่จอง วันหมดอายุการจอง
5. สมาชิก ประกอบไปด้วยรหัสสมาชิก ชื่อผู้ใช้งาน(User name) ชื่อจริง อีเมล์ รหัสผ่าน รวมไปถึงข้อมูลการยืมหนังสือและข้อมูลการจองหนังสือที่เกี่ยวข้อง
6. เจ้าหน้าที่ห้องสมุด ประกอบไปด้วยรหัสเจ้าหน้าที่ ชื่อผู้ใช้งาน ชื่อจริง อีเมล์ รหัสผ่าน
7. หน่วยควบคุมสิทธิในการเข้าถึงการใช้งาน ทำหน้าที่ในการควบคุมการเข้าถึงหน้าเว็บของผู้ใช้งานที่มีสิทธิที่ต่างกัน
8. หน่วยควบคุมการจัดการหนังสือ ช่วยในการจัดการหนังสือที่มีอยู่ในห้องสมุด
9. หน่วยควบคุมการค้นหาหนังสือ ช่วยในการค้นหาหนังสือโดยค้นหาตามเงื่อนไขที่ผู้ใช้งานกำหนด
10. หน่วยควบคุมการจัดการข่าวสาร ช่วยในการจัดการรายการข่าวสารที่ห้องสมุดต้องการจะประกาศให้ทราบ
11. หน่วยควบคุมการแสดงผลข่าวสาร ช่วยในการแสดงผลรายละเอียดข่าวสารและเก็บรายการข่าวสารล่าสุดที่ห้องสมุดประกาศไว้
12. หน่วยควบคุมการจัดการสมาชิก เพื่อช่วยในการจัดการสมาชิกที่มีอยู่ในห้องสมุด
13. หน่วยควบคุมการจัดการบรรณารักษ์ ช่วยในการจัดการรายชื่อบรรณารักษ์ที่มีอยู่ในห้องสมุด
14. หน่วยควบคุมการจัดการผู้ใช้งานส่วนบุคคล เพื่อช่วยในการจัดการข้อมูลต่างๆของผู้ใช้ และช่วยในการนำทางเจ้าหน้าที่ห้องสมุดไปยังส่วนใช้งานของเจ้าหน้าที่ห้องสมุดได้
15. หน่วยควบคุมการทำรายการยืม-คืนหนังสือ ช่วยจัดการกรณีที่เจ้าห้องห้องสมุดต้องการทำรายการยืม หรือ คืนหนังสือให้แก่ผู้ใช้งานห้องสมุด
16. หน่วยควบคุมการจองหนังสือและต่ออายุการยืมหนังสือ ช่วยจัดการในกรณีที่ผู้ใช้งานต้องการจองหนังสือและทำรายการยืมหนังสือต่อ

- **Application architecture**

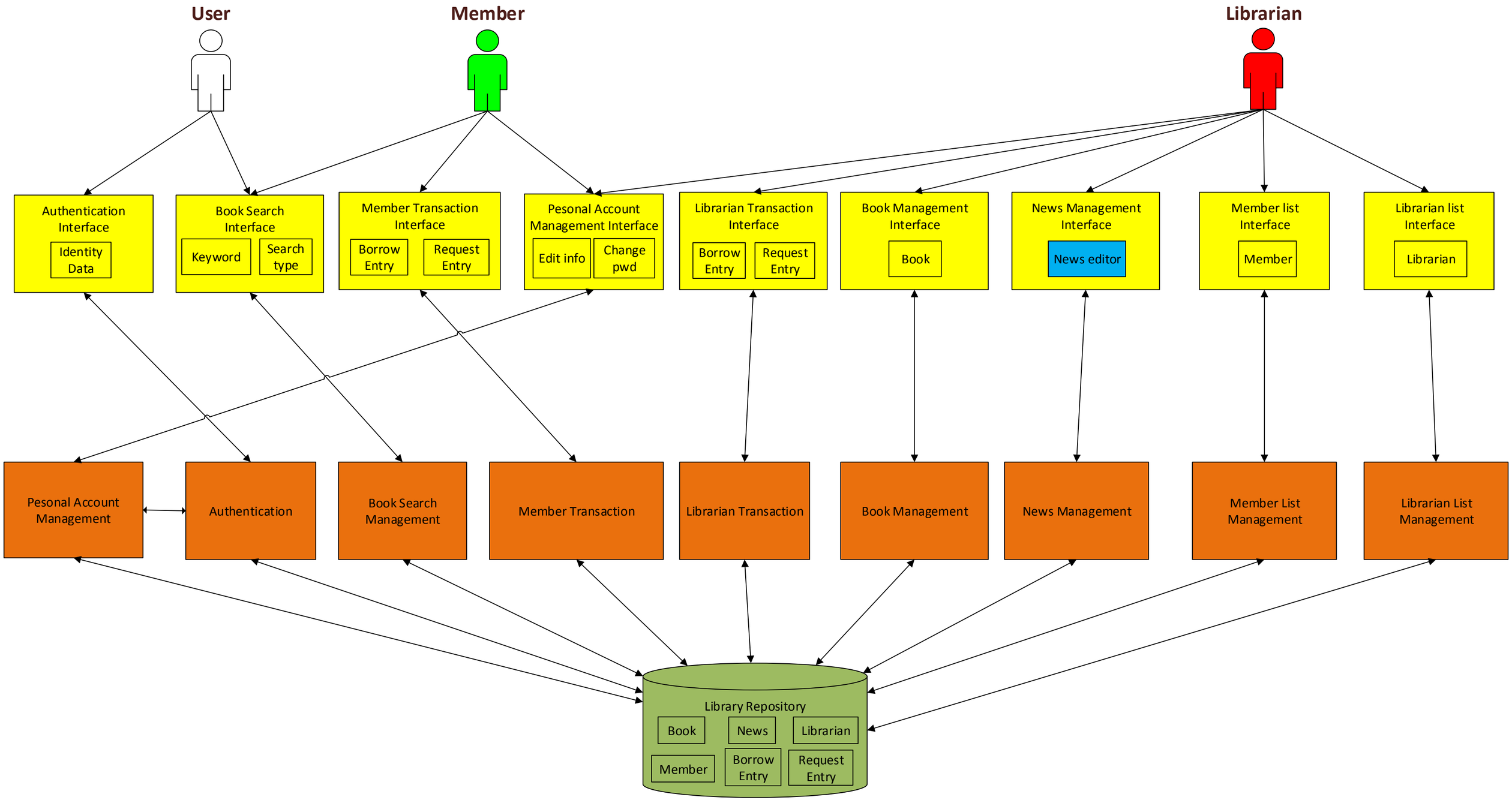
การพัฒนาระบบห้องสมุด ในส่วนของ Application architecture ได้ทำการแบ่งระบบออกเป็นส่วนๆ โดยมองจาก Abstraction และความสัมพันธ์กับ Functional requirement ซึ่งกลุ่มของระบบย่อยที่มีการทำงานคล้ายกันเช่น การเพิ่ม-แก้ไข-ลบ หนังสือ จะถูกจัดให้อยู่ในระบบเดียวกัน

จาก Abstraction ที่วิเคราะห์ได้ สามารถแบ่งออกเป็น Component ของระบบที่จะพัฒนาได้ดังนี้

1. ระบบจัดการหนังสือ ใช้เพื่อทำการเพิ่ม-ปรับปรุง-ลบข้อมูลหนังสือที่มีอยู่ในห้องสมุด โดยมีส่วนติดต่อกับผู้ใช้ในการส่ง-รับข้อมูลที่เกี่ยวกับรายละเอียดของหนังสือ
2. ระบบจัดการข่าวสาร ใช้เพื่อทำการเพิ่ม-ปรับปรุง-ลบข่าวสารที่ห้องสมุดต้องการประกาศ โดยมีส่วนติดต่อกับผู้ใช้ในการส่ง-รับข้อมูลที่เกี่ยวกับรายละเอียดของข่าวสาร
3. ระบบค้นหาหนังสือ ค้นหาหนังสือโดยค้นหาจากคำสำคัญที่ผู้ใช้งานให้มาเพื่อต้องการดูว่าหนังสือที่ต้องการค้นหามีอยู่จริงในห้องสมุดหรือไม่ โดยมีส่วนติดต่อกับผู้ใช้เพื่อรับข้อมูลที่เป็นคำสำคัญในการค้นหาและแสดงผลลัพธ์การค้นหา
4. ระบบทำรายการหนังสือสำหรับเจ้าหน้าที่ห้องสมุด ใช้เพื่อให้เจ้าหน้าที่ห้องสมุดสามารถทำรายการยืมหนังสือและทำรายการคืนหนังสือ รวมไปถึงทำการบันทึกข้อมูลการยืมลงในรายการยืม-คืนหนังสือ ซึ่งมีส่วนติดต่อกับผู้ใช้ในการทำรายการหนังสือและแสดงผลลัพธ์ของการทำรายการ
5. ระบบทำรายการหนังสือสำหรับสมาชิก ใช้เพื่อให้สมาชิกสามารถทำรายการที่เกี่ยวกับการต่ออายุการยืมและการจองหนังสือได้ โดยจะมีส่วนติดต่อผู้ใช้ในการรับข้อมูลของหนังสือที่จะต่ออายุหรือจองรวมไปถึงข้อมูลสมาชิกที่จะต่ออายุการยืมหรือจอง และแสดงผลลัพธ์ของการทำรายการ
6. ระบบจัดการสิทธิในการเข้าถึงการใช้งาน ทำหน้าที่ในการอนุญาต/จำกัด ผู้ใช้งานให้เข้าถึงในส่วนของระบบที่กำหนดไว้ได้ โดยอาจมีส่วนติดต่อกับผู้ใช้ในการแสดงผลสำหรับกรณีที่ระบบต้องการการยืนยันตัวตน
7. ระบบจัดการผู้ใช้งานส่วนบุคคล ช่วยให้ผู้ใช้งานที่เป็นสมาชิก/เจ้าหน้าที่ห้องสมุดสามารถจัดการข้อมูลส่วนตัวและสามารถดูข้อมูลอื่นๆที่เกี่ยวข้องกับตนเองได้ผ่านส่วนติดต่อผู้ใช้งาน
8. ระบบจัดการสมาชิกสำหรับเจ้าหน้าที่ห้องสมุด ทำหน้าที่ในการควบคุมสมาชิกในห้องสมุด สามารถลบสมาชิกในห้องสมุดออกได้ ผ่านส่วนติดต่อผู้ใช้ที่สามารถแสดงรายละเอียดและแสดงผลลัพธ์ของการดำเนินการเกี่ยวกับสมาชิก

สำหรับภาพรวมของ Application architecture แสดงในหน้าถัดไป

■ ภาพรวมของ Application architecture



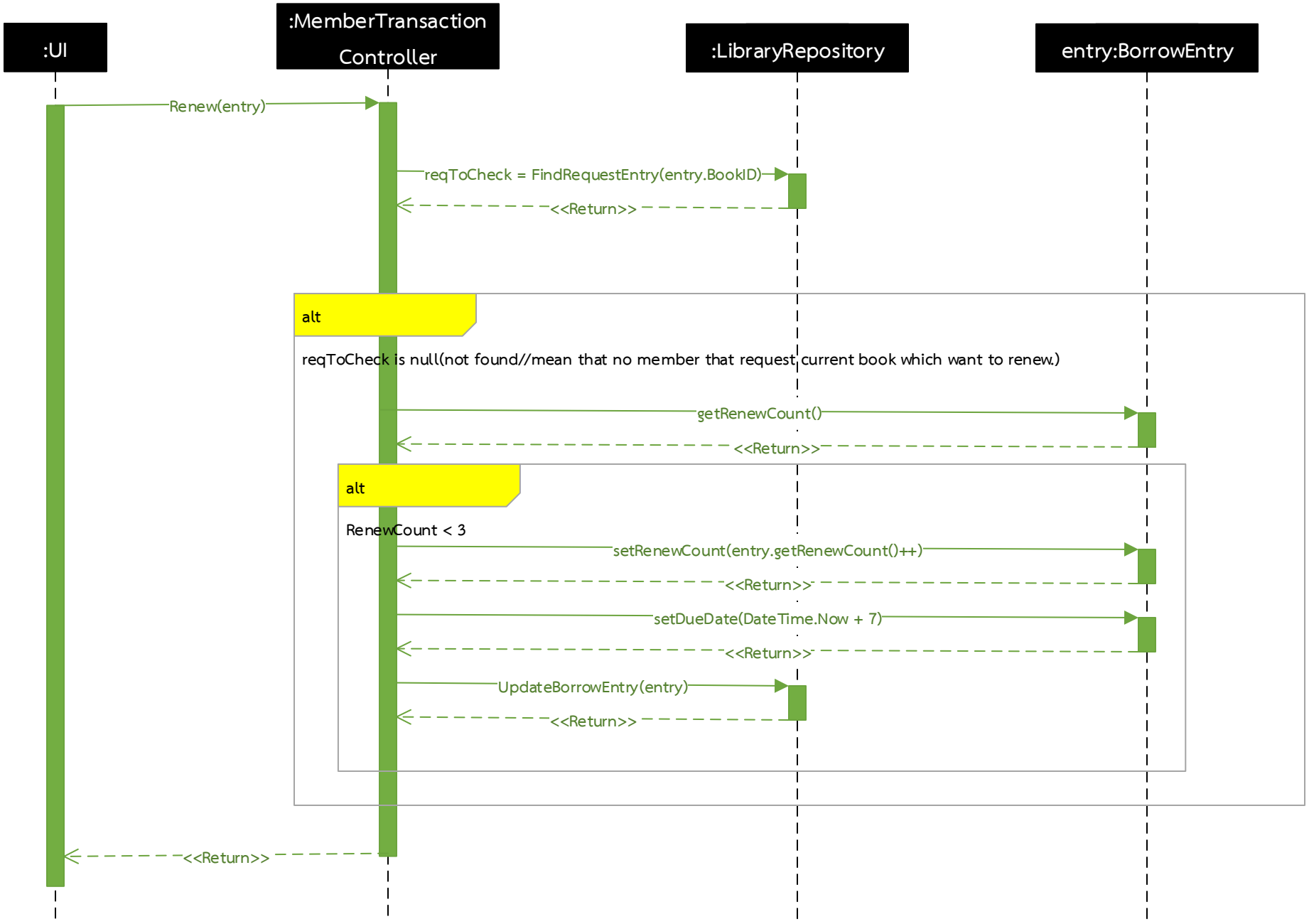
- **Subsystems / Components**

1. ระบบจัดการหนังสือ ทำหน้าที่ในการเพิ่ม-แก้ไข-ลบรายชื่อหนังสือโดยทำงานผ่านส่วนติดต่อผู้ใช้ที่สามารถส่งและรับรายละเอียดหนังสือได้
2. ระบบจัดการข่าวสาร ทำหน้าที่ในการเพิ่ม-แก้ไข-ลบรายการข่าวสารโดยทำงานผ่านส่วนติดต่อผู้ใช้ที่สามารถส่งและรับรายละเอียดข่าวสารได้
3. ระบบค้นหาหนังสือ ทำหน้าที่ในการค้นหาหนังสือ โดยระบบสามารถค้นหาได้ 3 แบบคือ แบบเร็ว แบบพื้นฐาน และแบบขั้นสูง โดยทำงานผ่านส่วนติดต่อผู้ใช้ที่สามารถส่งและรับรายละเอียดของการค้นหาได้
4. ระบบทำรายการหนังสือสำหรับเจ้าหน้าที่ห้องสมุด ทำหน้าที่ในการทำรายการยืมหนังสือ/คืนหนังสือ และติดตามการยืมหนังสือ โดยทำงานผ่านส่วนติดต่อผู้ใช้ที่สามารถรับและส่งรายละเอียดของสมาชิกและหนังสือที่ต้องการยืมหรือคืนได้
5. ระบบทำรายการหนังสือสำหรับสมาชิก ทำหน้าที่ในการทำรายการต่ออายุการยืม/จองหนังสือ และ ยกเลิกการจองหนังสือ โดยทำงานผ่านส่วนติดต่อผู้ใช้ที่สามารถรับและส่งรายละเอียดของสมาชิกและหนังสือที่ต้องการต่ออายุ/จองหรือยกเลิกการได้
6. ระบบจัดการสิทธิในการเข้าถึงการใช้งาน ทำหน้าที่ในการควบคุมสิทธิในการเข้าถึงหน้าในเว็บไซต์บางหน้าที่ต้องการให้สามารถเข้าถึงได้เฉพาะผู้ที่เข้าสู่ระบบแล้วเท่านั้น รวมไปถึงควบคุมเกี่ยวกับการสมัครสมาชิก การกู้คืนบัญชีผู้ใช้ โดยทำงานผ่านส่วนติดต่อผู้ใช้ในกรณีที่ใช้งานระบบสมัครสมาชิก/กู้คืนรหัสผ่าน/จะเข้าสู่ระบบ และทำงานร่วมกับระบบจัดการผู้ใช้งานส่วนบุคคลในการที่จะนำทางผู้ใช้งานที่เข้าสู่ระบบแล้วไปยังหน้าจัดการผู้ใช้งาน
7. ระบบจัดการผู้ใช้งานส่วนบุคคล ทำการที่ในการแสดงข้อมูลส่วนตัวของผู้ใช้งานแต่ละคน รวมไปถึงทำหน้าที่ในการแก้ไขข้อมูลผู้ใช้งานที่ผู้ใช้งานต้องการ(เช่น รหัสผ่าน,ชื่อ และ e-mail) โดยทำงานผ่านส่วนติดต่อผู้ใช้ที่สามารถรับและส่งรายละเอียดผู้ใช้งานได้
8. ระบบจัดการสมาชิกสำหรับเจ้าหน้าที่ห้องสมุด ทำหน้าที่ในการเรียกดูรายละเอียดของสมาชิกในห้องสมุด รวมไปถึงสมาชิกลบสมาชิกออกจากห้องสมุดได้ โดยทำงานผ่านส่วนติดต่อผู้ใช้ที่สามารถรับและแสดงผลข้อมูลสมาชิกได้
9. ระบบจัดการรายชื่อเจ้าหน้าที่ห้องสมุด ทำหน้าที่ในการเพิ่ม,ดูรายละเอียดและลบเจ้าหน้าที่ห้องสมุด โดยทำงานผ่านส่วนติดต่อผู้ใช้ที่สามารถรับและส่งข้อมูลเจ้าหน้าที่ห้องสมุดได้

Sequence diagram ที่จะนำเสนอในเอกสารนี้ทางกลุ่มนำเสนอ Sequence diagram ของกรณีทำรายการยืมหนังสือของเจ้าหน้าที่ห้องสมุด และ Sequence diagram ของกรณีการทำรายการต่ออายุการยืมหนังสือของสมาชิกซึ่งแสดงในหน้าถัดไป

Sequence Diagram สำหรับกรณีทำรายการยืมหนังสือของเจ้าหน้าที่ห้องสมุด

Sequence Diagram สำหรับกรณีการทำรายการต่ออายุการยืมหนังสือของสมาชิก



แผนภาพของคลาสหลัก

- **Domain class**

ในส่วนของ Business logic ได้ใช้แนวคิดของ Model-View-Controller ร่วมกับ repository pattern ในการนำเสนอ Abstraction โดยจะมี class ที่เป็น Model หลักๆก็คือ หนังสือ(Book),เจ้าหน้าที่ห้องสมุด(Librarian),สมาชิก(Member),ข่าวสาร (News),รายการยืม-คืนหนังสือ(BorrowEntry) และ รายการจองหนังสือ(RequestEntry)

สำหรับความสัมพันธ์ใน Model สามารถสรุปได้เป็นดังนี้

- Class Member และ Librarian ต่างก็สืบทอดมาจาก Abstract class Person ทั้งคู่ซึ่งมีการ Implement ฟังก์ชัน Identify() เอาไว้ใช้สำหรับแสดงผลว่า Object นั้นๆมีสถานะเป็นผู้ใช้งานประเภทใดจาก 1 ใน 2 ประเภทพร้อมทั้งแสดงชื่อผู้ใช้งานประกอบ
- Book และ BorrowEntry สองคลาสนี้มีความสัมพันธ์กันแบบ Weak association กล่าวคือภายในสองคลาสนี้ มี method ที่เอาไว้สำหรับเรียกดูข้อมูลที่เกี่ยวข้องของอีกคลาสซึ่งกันและกัน
- Book และ RequestEntry สองคลาสนี้มีความสัมพันธ์กันแบบ Weak association กล่าวคือภายในสองคลาสนี้ มี method ที่เอาไว้สำหรับเรียกดูข้อมูลที่เกี่ยวข้องของอีกคลาสซึ่งกันและกัน
- Member และ BorrowEntry สองคลาสนี้มีความสัมพันธ์กันแบบ Weak association กล่าวคือภายในสองคลาสนี้ มี method ที่เอาไว้สำหรับเรียกดูข้อมูลที่เกี่ยวข้องของอีกคลาสซึ่งกันและกัน
- Member และ RequestEntry สองคลาสนี้มีความสัมพันธ์กันแบบ Weak association กล่าวคือภายในสองคลาสนี้ มี method ที่เอาไว้สำหรับเรียกดูข้อมูลที่เกี่ยวข้องของอีกคลาสซึ่งกันและกัน

ในส่วนของ Controller ได้แบ่งตาม Abstraction ที่เกี่ยวข้องกับหน่วยควบคุมต่างๆ โดยมี class ดังนี้

- BookManagerController คลาสนี้เกี่ยวข้องกับระบบจัดการหนังสือ
- BookSearchController คลาสนี้เกี่ยวข้องกับระบบการค้นหาหนังสือ
- NewsManagerController คลาสนี้เกี่ยวข้องกับระบบจัดการข่าวสาร
- NewsController คลาสนี้เกี่ยวข้องกับการเรียกดูรายละเอียดข่าวสารและเก็บข่าวสารล่าสุดที่จะแสดงผล
- MemberTransactionController คลาสนี้เกี่ยวข้องกับระบบการทำรายการของสมาชิก(จอง-ต่ออายุการยืม)
- LibrarianTransactionController คลาสนี้เกี่ยวข้องกับระบบการทำรายการของเจ้าหน้าที่ห้องสมุด(ยืม-คืน)
- MemberListController คลาสนี้เกี่ยวข้องกับระบบจัดการสมาชิกสำหรับเจ้าหน้าที่ห้องสมุด
- LibrarianListController คลาสนี้เกี่ยวข้องกับระบบจัดการรายชื่อเจ้าหน้าห้องสมุดสำหรับเจ้าหน้าที่ห้องสมุด
- AuthenticateController คลาสนี้เกี่ยวข้องกับระบบจัดการสิทธิในการเข้าถึงการใช้งาน
- AccountController คลาสนี้เกี่ยวข้องกับระบบจัดการผู้ใช้งานส่วนบุคคล

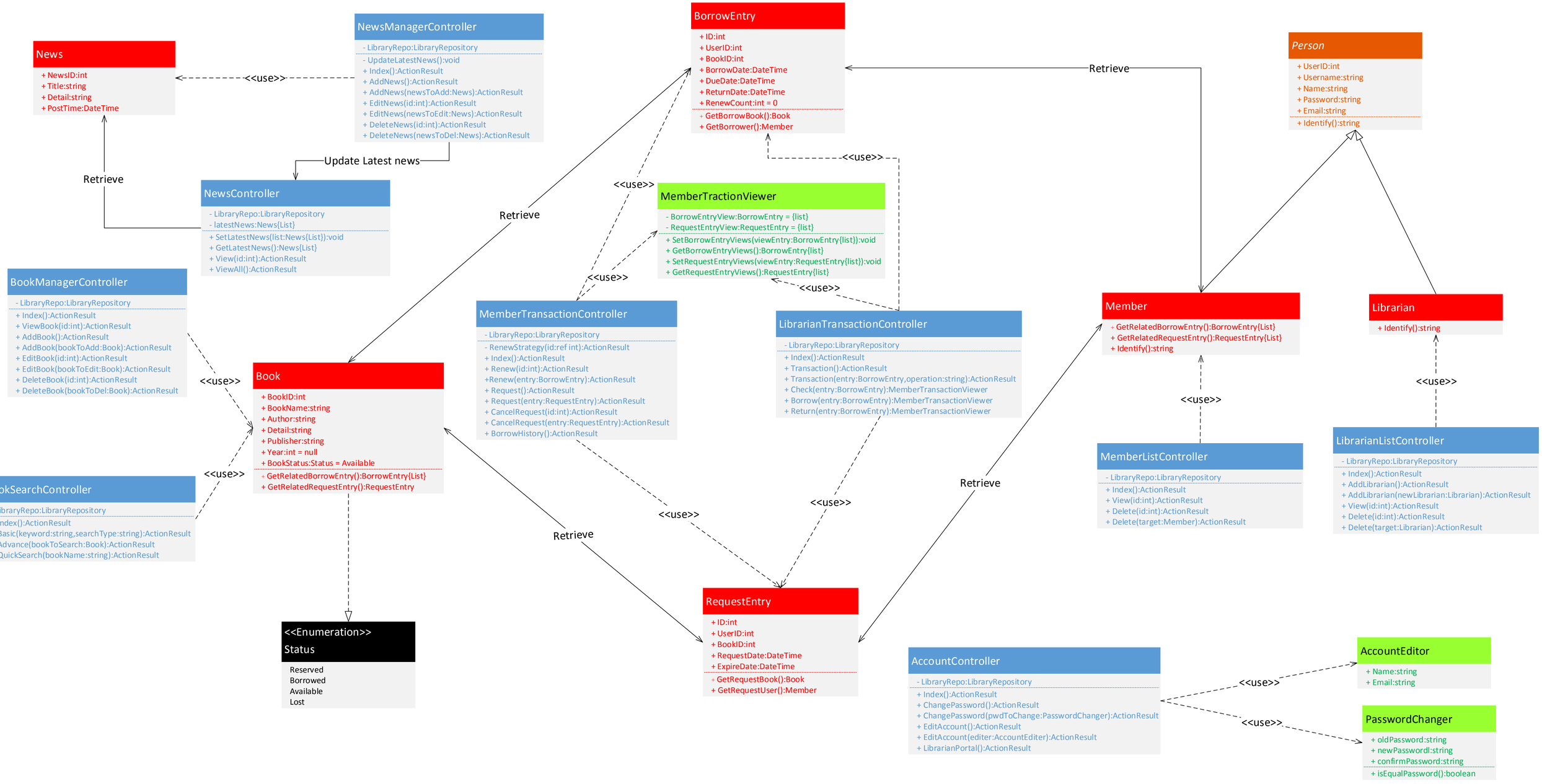
นอกจากนี้ยังมี class ที่เป็น ViewModel ซึ่งเป็นตัวช่วยในการรับข้อมูลบางอย่างจากผู้ใช้หรือแสดงผลข้อมูลบางอย่างจากหลายๆ Model รวมกัน ซึ่งเบื้องต้นมีคลาสในลักษณะนี้ 4 คลาสคือ

- MemberTractionViewer คลาสนี้มีหน้าที่สำหรับแสดงผลรายการยืมคืนและรายการจองพร้อมกันในเวลาเดียวกัน โดยใช้งานร่วมกันระบบการทำรายการของเจ้าหน้าที่ห้องสมุด-สมาชิก
- AccountEditor คลาสนี้ทำหน้าที่เป็นตัวรับข้อมูลจากผู้ใช้ในกรณีที่ผู้ใช้ต้องการแก้ไขข้อมูลส่วนตัวต่างๆ ยกเว้นรหัสผ่าน
- PasswordChanger คลาสนี้ทำหน้าที่เป็นตัวรับข้อมูลจากผู้ใช้ในกรณีที่ผู้ใช้ต้องการเปลี่ยนรหัสผ่าน
- LoginForm คลาสนี้ทำหน้าที่เป็นตัวรับข้อมูลจากผู้ใช้ในกรณีที่ผู้ใช้ต้องการกรอกข้อมูลยืนยันตัวตนเพื่อเข้าสู่ระบบ

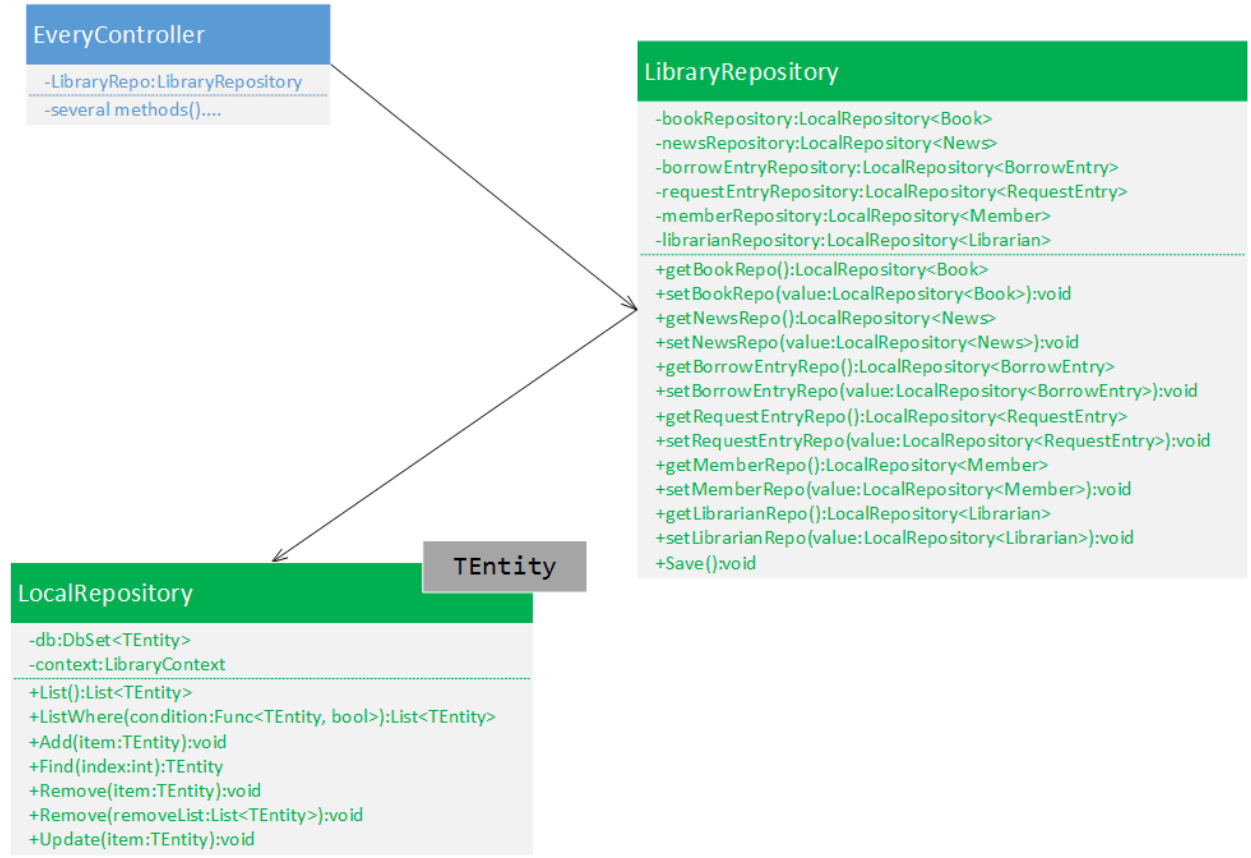
ภายใน Controller ต่างก็ไปใช้คลาสที่เป็น Model/ViewModel ต่างๆมาเป็นพารามิเตอร์ของ method มากมาย ดังจะเห็นได้จากแผนภาพ Class Diagram ที่ Controller หลายตัวมีความสัมพันธ์กับคลาสอื่นๆแบบ Dependency นอกจากความสัมพันธ์แบบ Dependency ที่เห็นได้ชัดแล้ว ใน class NewsManagerController และ NewsController มีความสัมพันธ์กันแบบ Weak association จาก class NewsManagerController ไปหา NewsController เพราะ class NewsManager นั้นมีการเรียกใช้ method UpdateLatestNews ซึ่งเอาไว้สำหรับตั้งค่ารายการข่าวสารล่าสุดซึ่งถูกเรียกหลังจากการเพิ่ม/แก้ไขหรือลบข่าวสาร โดยภายใน method ดังกล่าวได้มีการเรียก method ที่เป็น static ของ class NewsController ที่ชื่อ SetLatestNews เพื่อทำการตั้งรายการข่าวสารล่าสุดจริงๆ สำหรับตัว class NewsController เองก็มีความสัมพันธ์อีก 1 ต่อกับ class News โดยมีความสัมพันธ์แบบ Weak association จาก class NewsController ไปหา News เพราะในคลาส NewsController มี method ที่เอาไว้สำหรับเรียกดูรายการข่าวสารล่าสุดซึ่งจะคืนค่าออกมาเป็นข้อมูลของ class News

สำหรับการออกแบบ Domain class ในภาพรวมสำหรับส่วนไม่ใช่ persistence จาก Class diagram ในหน้าถัดไป คลาสที่เป็นสีแดงหมายถึง Model คลาสที่เป็นสีส้มหมายถึง Abstract class คลาสที่เป็นสีฟ้าหมายถึง Controller ส่วนคลาสที่เป็นสีเขียวหมายถึง ViewModel

■ Class diagram of domain class/business logic(Main module)



อีกส่วนหนึ่งที่ยังไม่ได้กล่าวถึงคือส่วนที่เป็น persistence ส่วนนี้ใช้แนวคิด Repository pattern ในการออกแบบคลาสที่เป็นตัวกลางในการเชื่อมต่อกับฐานข้อมูลอีกต่อหนึ่ง โดยหลักๆแล้วคลาสที่เป็น controller ทุกคลาสจะมี Object ที่เป็น repository เป็น 1 ใน properties ของ controller เพื่อที่ controller จะได้เรียกใช้การทำงานที่เกี่ยวข้องกับการเชื่อมต่อฐานข้อมูลผ่าน object ดังกล่าวได้ ดังนั้น ความสัมพันธ์ระหว่างคลาส controller กับคลาสที่เป็น repository ซึ่งในที่นี้คือ LibraryRepository มีความสัมพันธ์เป็นแบบ Direct association จาก class ที่เป็น controller ไปหา LibraryRepository แต่เนื่องจากว่าข้อมูลที่เก็บในระบบนั้นไม่ได้มีเพียงแค่ข้อมูลชนิดเดียว ซึ่งระบบนั้นจะต้องเก็บข้อมูลถึง 6 ชนิดด้วยกัน(เก็บข้อมูลของ Model ทั้งหมด) เลยเป็นสาเหตุที่ทำให้ภายในคลาส LibraryRepository มี LocalRepository ที่ bind กับ class ที่เป็น Model ทั้ง 6 แน่นอนว่าความสัมพันธ์ระหว่างคลาส LibraryRepository กับ LocalRepository ก็เป็นแบบ Direct association(จาก class LibraryRepository ไปหา LocalRepository) เช่นกัน สำหรับ class diagram ในส่วนที่เป็น persistence นั้นแสดงดังภาพด้านล่าง



<u>รายละเอียดการพัฒนาซอฟต์แวร์</u>

- **<u>Deployment</u>**

สำหรับการพัฒนาระบบเพื่อใช้งานจริง ในฝั่ง Server-side จะใช้ VM จำลอง Windows Server 2012 R2 โดยมี IIS เป็นตัวจัดการ Web Server และใช้ Framework ASP.NET MVC เป็นเครื่องมือในการพัฒนา ในส่วนของ API จะมี Framework ASP.NET Web API ในการพัฒนาด้วยเช่นกัน ขณะที่ Persistence(Database) จะใช้ Entity Framework ในการจัดการร่วมกับ SQL Database คุณสมบัติของระบบทั้งหมดใน Application architecture ยกเว้นส่วนติดต่อผู้ใช้ จะทำงานบน VM ดังที่ได้อธิบายไว้ สำหรับฝั่ง Client-side ใช้ JQuery/JQueryUI ในการจัดการส่วนติดต่อผู้ใช้และการแสดงผลต่างๆ ซึ่งระบบจัดการข่าวสารจะมีการใช้ CKEditor ในการเพิ่ม/แก้ไขข่าวสารที่สามารถเก็บข้อมูลเป็น HTML mark up ได้อีกด้วย VM Server IP ที่ใช้งานในระบบนี้ คือ 168.63.172.78 (domain name: http://paratabplus.cloudapp.net)

- **<u>Implementation plan</u>**

ลำดับ	งานที่ดำเนินการ	ผู้รับผิดชอบ	ระยะเวลา
1	ศึกษาและติดตั้ง IIS บน VM ฝึกการใช้งาน ASP.NET MVC/Web API และโครงสร้างไดเรคทอรี่ที่ใช้สำหรับทำงาน	สุรวิทย์, สุรัชนันท์	1 กันยายน 2557 - 7 กันยายน 2557
2	ออกแบบเค้าโครง Layout ของหน้าเว็บ และ CSS Style	สุรัชนันท์	8 กันยายน 2557 - 14 กันยายน 2557
3	ออกแบบส่วนของแสดงผลหน้าเว็บทั่วไป (Home/About us/etc.)	สุรัชนันท์	15 กันยายน 2557 – 21 กันยายน 2557
4	ออกแบบและสร้าง Database	สุรวิทย์	22 กันยายน 2557 – 23 กันยายน 2557
5	Implement Class ที่เกี่ยวข้องกับสมาชิกและเจ้าหน้าที่ห้องสมุด	สุรวิทย์, สุรัชนันท์	24 กันยายน 2557 – 26 กันยายน 2557
6	Implement ระบบสมัครสมาชิก พร้อมทดสอบ	สุรวิทย์, สุรัชนันท์	27 กันยายน 2557 – 30 กันยายน 2557
7	Implement ระบบ login พร้อมทดสอบ	สุรวิทย์, สุรัชนันท์	1 ตุลาคม 2557 – 3 ตุลาคม 2557
8	Implement ระบบจัดการหนังสือ(เพิ่ม/แก้ไข/ลบ) พร้อมทดสอบ	สุรวิทย์, สุรัชนันท์	4 ตุลาคม 2557 – 10 ตุลาคม 2557
9	Implement ระบบการค้นหาหนังสือ และแสดงรายละเอียดหนังสือ พร้อมทดสอบ	สุรวิทย์, สุรัชนันท์	11 ตุลาคม 2557 – 15 ตุลาคม 2557
10	Implement	สุรวิทย์	16 ตุลาคม 2557 –

	ระบบกำหนดสิทธิการเข้าถึง พร้อมทดสอบ		18 ตุลาคม 2557
11	ทดสอบการทำงานของระบบทั้งหมดครั้งที่ 1	สุรวิทย์	19 ตุลาคม 2557
12	Implement ระบบทำรายการยืม-คืนหนังสือ พร้อมทดสอบ	สุรัชนันท์	20 ตุลาคม 2557 – 25 ตุลาคม 2557
13	Implement ระบบจัดการสมาชิกของห้องสมุด พร้อมทดสอบ	สุรวิทย์	26 ตุลาคม 2557 – 27 ตุลาคม 2557
14	ทดสอบการทำงานของระบบทั้งหมดครั้งที่ 2	สุรวิทย์	28 ตุลาคม 2557
15	Implement ระบบต่ออายุการยืมหนังสือ พร้อมทดสอบ	สุรวิทย์,สุรัชนันท์	29 ตุลาคม 2557 – 31 ตุลาคม 2557
16	Implement ระบบทำรายการจองหนังสือ พร้อมทดสอบ	สุรวิทย์,สุรัชนันท์	1 พฤศจิกายน 2557 – 3 พฤศจิกายน 2557
17	Implement ระบบจัดการข้อมูลส่วนตัวของผู้ใช้ พร้อมทดสอบ	สุรวิทย์	4 พฤศจิกายน 2557 – 6 พฤศจิกายน 2557
18	Implement ระบบจัดการข่าวสาร พร้อมทดสอบ	สุรัชนันท์	7 พฤศจิกายน 2557 – 9 พฤศจิกายน 2557
19	Implement ทดสอบการทำงานของระบบทั้งหมดครั้งสุดท้าย	สุรวิทย์	10 พฤศจิกายน 2557 – 12 พฤศจิกายน 2557
20	ทดสอบความเสถียรของระบบ พร้อมแก้ไข จุดบกพร่อง	สุรวิทย์,สุรัชนันท์	13 พฤศจิกายน 2557 – 16 พฤศจิกายน 2557

ผลการทดสอบซอฟต์แวร์

- **Test**

  การทดสอบที่จะใช้ในการวัดว่าระบบทำงานได้ถูกต้องหรือไม่ ทางกลุ่มได้หยิบยก 4 การทดสอบที่จะใช้ในการทดสอบระบบที่พัฒนาขึ้นโดยแบ่งการประเภททดสอบออกเป็น Unit test และ Integration test ดังนี้

  - **Unit test**

    1. ทดสอบการทำงานของระบบการต่ออายุการยืมและการจอง/ยกเลิกการจองหนังสือ
       ในการทดสอบนี้ได้มีการทดสอบการทำรายการของสมาชิก(ต่ออายุการยืม/การจอง/ยกเลิกการจอง) ในรูปแบบเงื่อนไขต่างๆที่สามารถเกิดขึ้นได้ ก่อนการทดสอบได้มีการเตรียมข้อมูลสำหรับทดสอบ คือ ข้อมูลสมาชิก 5 คน,หนังสือ 7 เล่ม,รายการยืมหนังสือ 7 รายการ และรายการจองหนังสือ 2 รายการ รวมไปถึงคลาสจำเป็นบางคลาสที่ต้องมีการ mock เพื่อทำให้ test method สามารถทำงานได้ โดยกรณีการทดสอบที่ได้หยิบยกมาอธิบาย มีดังนี้

       1.1 ทดสอบการทำการของการเรียกหน้ายืนยันการต่ออายุ กรณีปกติ

```
[TestMethod]
0 references
public void TestRenewAction1()
{
    InitialController("M_ce51benz", libRepo.Object);
    ViewResult result = controller.Renew(1) as ViewResult;
    Assert.IsNotNull(result.Model as BorrowEntry);
    Assert.AreEqual(1, (result.Model as BorrowEntry).ID);
    Assert.AreEqual(1, (result.Model as BorrowEntry).BookID);
    Assert.AreEqual(2, (result.Model as BorrowEntry).UserID);
    Assert.IsNull(result.TempData["ErrorNoti"]);
}
```

สมาชิกที่มีการยืมหนังสือไว้และยังไม่ได้คืน สามารถต่ออายุการยืมได้ ถ้าหนังสือที่ยืมยังไม่เกินกำหนดคืน และรายการต่ออายุที่เรียกต้องเป็นรายการยืมของตนเองเท่านั้น สำหรับโค้ดที่ทำการทดสอบนั้น ได้ทำการทดสอบกับสมาชิก ce51benz โดยทำการทดสอบง่ายๆ คือ หลังจากเรียกหน้าต่ออายุแล้ว ผลลัพธ์ที่คาดไว้ก็คือหน้าข้อมูลการยืนยันการต่ออายุ มีข้อมูลหนังสือ เลขหนังสือ และเลขผู้ใช้งานที่ต่ออายุ ตรงตามที่กำหนดไว้ในข้อมูลทดสอบจริงๆ

1.2 ทดสอบการเรียกหน้ายืนยันการต่ออายุ กรณีต่ออายุหนังสือที่ยืมเกินกำหนด

```
[TestMethod]
0 references
public void TestRenewAction5()
{
    InitialController("M_baybaybay", libRepo.Object);
    RedirectToRouteResult result = controller.Renew(7) as
        RedirectToRouteResult;
    Assert.AreEqual("Index", result.RouteValues["action"]);
    Assert.AreEqual("Cannot renew this book due to borrow overdue.",
        controller.TempData["ErrorNoti"]);
}
```

ในการทดสอบนี้ ได้กำหนดให้สมาชิก baybaybay พยายามที่จะเรียกหน้าต่ออายุการยืมหนังสือเพื่อดำเนินการต่ออายุให้ได้ ซึ่งหนังสือที่ตัวเองต้องการต่ออายุนั้นเกินกำหนดการคืนแล้ว ผลลัพธ์ที่คาดหวังจากการทดสอบกรณีนี้คือ TempData[“ErrorNoti”] จะเซตข้อความแจ้งกลับไปยังสมาชิกว่า Cannot renew this book due to borrow overdue.

1.3 ทดสอบการเรียกหน้ายืนยันการต่ออายุ กรณีหนังสือที่ยืมนั้นต่ออายุครบจำนวนครั้งที่กำหนด

```
[TestMethod]
0 references
public void TestRenewAction4()
{
    InitialController("M_tomza2014", libRepo.Object);
    RedirectToRouteResult result = controller.Renew(6) as
        RedirectToRouteResult;
    Assert.AreEqual("Index", result.RouteValues["action"]);
    Assert.AreEqual("Your renew of book ID.6 is exceed maximum!",
        controller.TempData["ErrorNoti"]);
}
```

ในการทดสอบนี้ ได้กำหนดให้สมาชิก tomza2014 พยายามที่จะต่ออายุเป็นครั้งที่ 4 ซึ่งผิดไปจากกฎที่ตั้งไว้ ผลลัพธ์ที่คาดหวังจากการทดสอบนี้คือ TempData[“ErrorNoti”] จะเซตข้อความแจ้งกลับไปยังสมาชิกว่า Your renew of book ID.6 is exceed maximum!

1.4 ทดสอบการทำรายการต่ออายุการยืมหลังจากกดยืนยันการต่ออายุจริงๆ (สำหรับกรณีปกติ)

```
[TestMethod]
0 references
public void TestRenewPostAction2()
{
    InitialController("M_ce51benz", libRepo.Object);
    RedirectToRouteResult result = controller.Renew(libRepo.Object.
        BorrowEntryRepo.ListWhere
        (target => target.ID == 4).Single()) as RedirectToRouteResult;
    Assert.AreEqual("Index", result.RouteValues["action"]);
    Assert.IsNull(controller.ControllerContext.Controller.TempData["ErrorNoti"]);
    Assert.AreEqual("Renew successful!", controller.ControllerContext.
        Controller.TempData["SuccessNoti"]);
}
```

ในกรณีปกติ ถ้าสมาชิกผู้ที่ต้องการต่ออายุการยืมของหนังสือ ได้กดยืนยันไปแล้ว สิ่งที่คาดหวังจากกรณีนี้คือ มีข้อความแจ้งเตือนที่เซ็ตผ่าน TempData["SuccessNoti"] ว่า Renew successful! โดยกรณีทดสอบนี้ทำกับสมาชิก ce51benz

1.5 ทดสอบการทำรายการต่ออายุการยืมหลังจากกดยืนยันการต่ออายุการยืมของหนังสือ (กรณีไม่สำเร็จ)

```
[TestMethod]
0 references
public void TestRenewPostAction1()
{
    InitialController("M_ce51benz", libRepo.Object);
    RedirectToRouteResult result = controller.Renew(libRepo.Object.
        BorrowEntryRepo.ListWhere
        (target => target.ID == 1).Single()) as RedirectToRouteResult;
    Assert.AreEqual("Index", result.RouteValues["action"]);
    Assert.IsNull(controller.ControllerContext.Controller.
        TempData["SuccessNoti"]);
    Assert.AreEqual("This book is ON HOLD.", controller.
        ControllerContext.Controller.TempData["ErrorNoti"]);
}
```

ในกรณีนี้ หนังสือที่ต้องการต่ออายุการยืมนั้น ได้มีสมาชิกคนอื่นจองไว้แล้วซึ่งแน่นอนว่าสมาชิกที่ยืมหนังสือคนปัจจุบันก็ไม่สามารถต่ออายุการยืมของหนังสือเล่มนั้นได้ ในหน้ายืนยันการต่ออายุการยืมของหนังสือ ไม่ได้ตรวจสอบเงื่อนไขนี้ แต่มาตรวจสอบจริงๆหลังจากกดยืนยันแล้ว สิ่งที่คาดหวังจากการทดสอบกรณีนี้คือ คือ TempData[“ErrorNoti”] จะเซตข้อความ error แจ้งกลับไปยังสมาชิกว่า This book is ON HOLD. โดยกรณีทดสอบนี้ทำกับสมาชิก ce51benz

1.6 ทดสอบกรณีทำรายการจองหนังสือ – กรณีปกติ

```
[TestMethod]
0 references
public void TestRequestPostAction1()
{
    InitialController("M_baybaybay", libRepo.Object);
    RequestEntry postReq = new RequestEntry { BookID = 3 };
    RedirectToRouteResult result = controller.Request(postReq) as RedirectToRouteResult;
    Assert.AreEqual("Index", result.RouteValues["action"]);
    Assert.AreEqual("Request book successfully.", controller.
        ControllerContext.Controller.TempData["SuccessNoti"]);
    Assert.IsNull(controller.ControllerContext.Controller.TempData["ErrorNoti"]);
}
```

ในกรณีทดสอบนี้ สมมติให้สมาชิก baybaybay ทำการจองหนังสือที่มีคนอื่นยืมอยู่ และยังไม่มีใครจอง ผลลัพธ์ที่คาดหวังไว้จากการทดสอบนี้คือ TempData["SuccessNoti"] แสดงความข้อความแจ้งว่า Request book successfully.

1.7 ทดสอบกรณีทำรายการจองหนังสือ – กรณีจองหนังสือที่มีคนอื่นจองอยู่แล้ว

```
[TestMethod]
0 references
public void TestRequestPostAction4()
{
    InitialController("M_fonniz", libRepo.Object);
    RequestEntry postReq = new RequestEntry { BookID = 2 };
    ViewResult result = controller.Request(postReq) as ViewResult;
    Assert.AreEqual("This book is already requested.",
        result.TempData["ErrorNoti"]);
    Assert.IsNull(controller.ControllerContext.
        Controller.TempData["SuccessNoti"]);
}
```

ในกรณีทดสอบนี้ สมมติให้สมาชิก fonniz ทำการจองหนังสือที่มีคนอื่นจองไว้อยู่แล้ว ผลลัพธ์ที่คาดหวังไว้จากการทดสอบนี้คือ TempData[“ErrorNoti”] แสดงข้อความแจ้ง error กลับมายังสมาชิกว่า This book is already requested.

1.8 ทดสอบกรณีทำรายการจองหนังสือ – กรณีหนังสือที่ต้องการจองไม่ได้อยู่ในสถานะที่ถูกยืม

```
[TestMethod]
0 references
public void TestRequestPostAction6()
{
    InitialController("M_paratab", libRepo.Object);
    RequestEntry postReq = new RequestEntry { BookID = 4 };
    ViewResult result = controller.Request(postReq) as ViewResult;
    Assert.AreEqual("Can't request this book due to it is Available.",
        result.TempData["ErrorNoti"]);
    Assert.IsNull(controller.ControllerContext.Controller.
        TempData["SuccessNoti"]);
}
```

ในกรณีทดสอบนี้ สมมติให้สมาชิก paratab ทำการจองหนังสือเล่มที่ไม่มีใครยืม สถานะ Available ผลลัพธ์ที่คาดหวังไว้จากการทดสอบนี้คือ คือ TempData[“ErrorNoti”] แสดงข้อความแจ้ง error กลับมายังสมาชิกว่า Can't request this book due to it is Available.

1.9 ทดสอบการเรียกหน้ายืนยันการยกเลิกการจองหนังสือ – กรณีปกติ

```
[TestMethod]
0 references
public void TestCancelRequestAction3()
{
    InitialController("M_tomza2014", libRepo.Object);
    ViewResult result = controller.CancelRequest(2) as ViewResult;
    Assert.IsNull(controller.ControllerContext.Controller.TempData["ErrorNoti"]);
    Assert.IsNull(controller.ControllerContext.Controller.TempData["SuccessNoti"]);
    Assert.AreEqual(2, (result.Model as RequestEntry).BookID);
}
```

ในกรณีทดสอบนี้ กำหนดให้สมาชิก tomza2014 ทำการเรียกหน้ายกเลิกการจองหนังสือที่ตนเองเคยจองไว้ โดยผลลัพธ์ที่คาดหวังไว้จากการทดสอบนี้นั้น วัดได้ง่ายๆ โดยวัดว่า มีข้อมูลรายการจองหนังสือที่ต้องการยกเลิกการจองส่งกลับไปยังสมาชิกเป็นรายการที่คาดหวังไว้หรือไว้

1.10 ทดสอบการเรียกหน้ายืนยันการยกเลิกการจองหนังสือ – กรณีจะ Hack หรือพยายามไปเรียกหน้ายกเลิกการจองหนังสือของคนอื่น

```
[TestMethod]
0 references
public void TestCancelRequestAction2()
{
    InitialController("M_paratab", libRepo.Object);
    RedirectToRouteResult result = controller.CancelRequest(2) as RedirectToRouteResult;
    Assert.AreEqual("Can't cancel other member's book request.",
        controller.ControllerContext.Controller.TempData["ErrorNoti"]);
    Assert.IsNull(controller.ControllerContext.Controller.TempData["SuccessNoti"]);
}
```

ในกรณีทดสอบนี้ กำหนดให้สมาชิก paratab พยายามที่จะ hack โดยกด browse ไปที่หน้ายกเลิกการจองหนังสือที่ไม่ใช่ของตนเอง ผลลัพธ์ที่คาดหวังไว้จากการทดสอบนี้คือ คือ TempData[“ErrorNoti”] แสดงข้อความแจ้ง error กลับมายังสมาชิกว่า Can't cancel other member's book request.

1.11 ทดสอบการทำรายการยกเลิกการจองหนังสือหลังจากกดยืนยันแล้ว – กรณีปกติ

```
[TestMethod]
0 references
public void TestCancelRequestPostAction1()
{
    InitialController("M_tomza2014", libRepo.Object);
    RedirectToRouteResult result = controller.CancelRequest
        (new RequestEntry { BookID = 2 }) as RedirectToRouteResult;
    Assert.AreEqual("Index", result.RouteValues["action"]);
    Assert.IsNull(controller.ControllerContext.Controller.TempData["ErrorNoti"]);
    Assert.AreEqual("Cancel request successfully.",
        controller.ControllerContext.Controller.TempData["SuccessNoti"]);
}
```

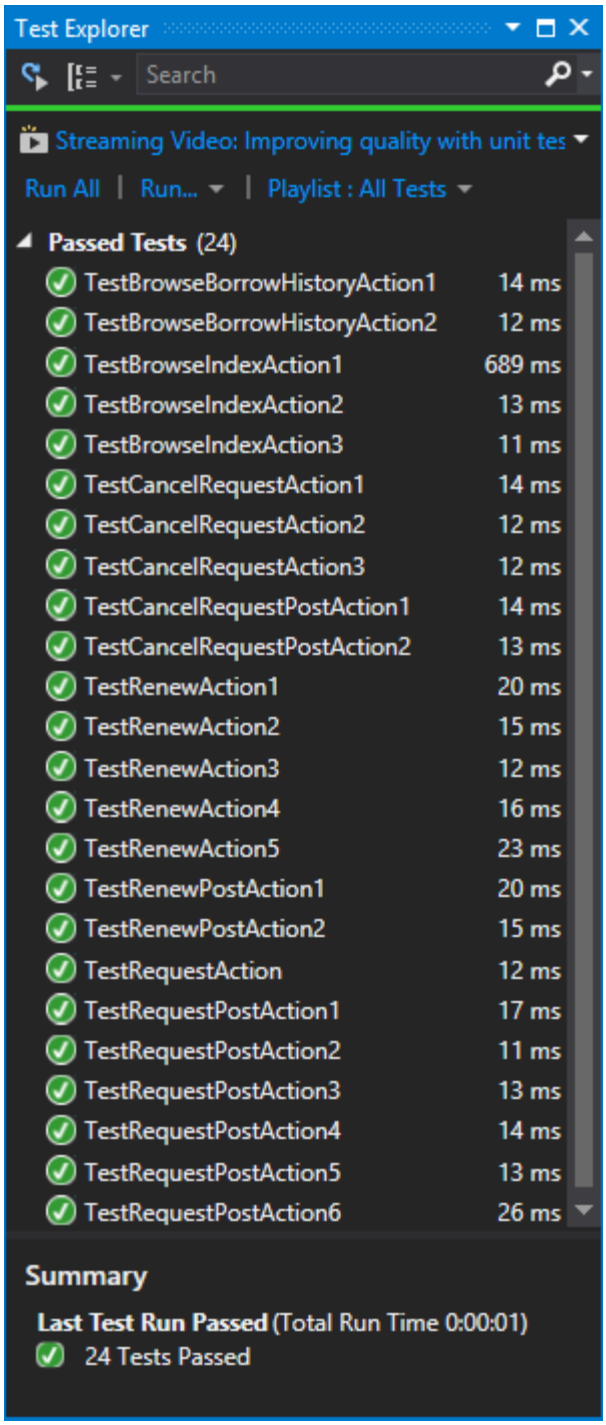
ในกรณีทดสอบนี้ สมาชิก tomza2014 ต้องการที่จะยกเลิกการจองหนังสือที่เคยจองจริงๆ ก็กดยืนยันไป ผลลัพธ์ที่คาดหวังไว้จากการทดสอบนี้คือ มีข้อความแสดงการทำรายการสำเร็จผ่าน TempData["SuccessNoti"] ว่า Cancel request successfully.

1.12 ทดสอบการทำรายการยกเลิกการจองหนังสือหลังจากกดยืนยันแล้ว – กรณีผิดพลาด

```
[TestMethod]
0 references
public void TestCancelRequestPostAction2()
{
    InitialController("M_tomza2014", libRepo.Object);
    RedirectToRouteResult result = controller.CancelRequest
        (new RequestEntry { BookID = 222 }) as RedirectToRouteResult;
    Assert.AreEqual("Index", result.RouteValues["action"]);
    Assert.IsNull(controller.ControllerContext.Controller.TempData["SuccessNoti"]);
    Assert.AreEqual("Something went wrong.", controller.
        ControllerContext.Controller.TempData["ErrorNoti"]);
}
```

ในกรณีทดสอบนี้ สมาชิกคนเดิมจากข้อที่แล้ว ต้องกดทำรายการยกเลิกการจองจริงๆ แต่เกิดความผิดพลาดในระหว่างการส่งข้อมูลรายการที่จะยกเลิกกลับไปหรือคนที่จะยกเลิกดันแก้ไขข้อมูลรายการการจอง ก็จะทำให้ระบบฟ้องกลับมาว่ามีบางอย่างผิดปกติขึ้น สิ่งที่คาดหวังไว้จากการทดสอบนี้คือ มีข้อความแจ้ง error กลับไปยังผู้ใช้ผ่าน TempData[“ErrorNoti”] ว่า Something went wrong.

สำหรับผลลัพธ์การทดสอบสำหรับกรณีทดสอบที่ยกมา และกรณีที่ไม่ได้ยกมา ผลการทดสอบคือผ่านทั้งหมด โดยผลการทดสอบนั้นเป็นไปตามภาพในหน้าถัดไป



2. ทดสอบการทำงานของระบบค้นหาหนังสือ

   ในการทดสอบนี้มีการกำหนด test case ที่อยู่ใน test method ที่เกี่ยวกับกรณีการค้นหาหนังสือในรูปแบบต่างๆ ซึ่งในแต่ละกรณีการทดสอบจะกำหนดรูปแบบการค้นหาที่แตกต่างกัน ทั้งกำหนด keyword ที่คาดว่าจะพบหนังสือที่ต้องการหรือไม่พบเลย และถ้าพบจะพบกี่รายการ รวมไปถึงตรวจสอบอย่างละเอียดว่าหนังสือที่พบนั้นมีผู้แต่งหรือข้อมูลอื่นๆ ตามที่คาดเดาไว้หรือไม่ ในกรณี

การทดสอบทางตรงกันข้ามก็มีการกำหนด input ที่ไม่ถูกต้องใส่เข้าไปในการค้นหา เช่น ค้นหาหนังสือตามปีที่พิมพ์ซึ่งต้องรับตัวเลขแต่ใส่เป็นตัวหนังสือธรรมดาไปแทน ซึ่งก็มีการคาดเดาเช่นเดียวกันว่าระบบจะฟ้อง error output กลับมายังผู้ใช้ให้ทำการกรอกเป็นรูปแบบที่ถูกต้อง ก่อนเริ่มต้นการทดสอบจะมีการเซตข้อมูลสำหรับทดสอบเอาไว้ด้วยเช่นกันโดยเป็นข้อมูลหนังสือจำนวน 30 เล่ม และรวมไปถึงคลาสจำเป็นบางคลาสที่ต้องมีการ mock เพื่อทำให้ test method สามารถทำงานได้ โดยกรณีการทดสอบที่ได้หยิบยกมาอธิบาย มีดังนี้

2.1 กรณีค้นหาหนังสือแบบค้นหาโดยผู้แต่งอย่างเดียวซึ่งผู้แต่งที่ต้องการค้นหาชื่อ J.K. Rowling โค้ดสำหรับทดสอบคือ

```
[TestMethod]
0 references
public void TestBasicSearchAction8()
{
    InitialController("M_ce51benz", libRepo.Object);
    ViewResult result = controller.Basic("J.K. Rowling", "Author", 1, 10) as ViewResult;
    Assert.AreEqual(7, (result.Model as PageList<Book>).GetList().Count);
    Assert.AreEqual("J.K. Rowling", result.TempData["Keyword"]);
    Assert.AreEqual(2002, (result.Model as PageList<Book>).GetList().
        Where(target => target.BookName == "Harry Potter and the Order of the Phoenix")
        .Single().Year);
    Assert.AreEqual(2005, (result.Model as PageList<Book>).GetList().
    Where(target => target.BookName == "Harry Potter and the Half-Blood Prince")
        .Single().Year);
    Assert.AreEqual(7, (result.Model as PageList<Book>).GetList().
        Where(target => target.Publisher == "Bloomsbury Publishing").ToList().Count);
    Assert.IsNull(result.TempData["ErrorNoti"]);
}
```

ภายในโค้ดทดสอบนั้นมีการเรียกใช้ method Basic ของ class BookSearchController ซึ่งจะค้นหาหนังสือตามชนิดและ keyword ที่ต้องการค้นหา โดยเมื่อทำการส่งผ่านพารามิเตอร์โดยมีชื่อผู้แต่ง J.K. Rowling และรูปแบบการค้นหาเป็นค้นหาโดยผู้แต่ง (จาก code คือพารามิเตอร์ตัวถัดไป => “author”) สิ่งที่คาดหวังจากการเรียก method นี้คือ พบผลลัพธ์การค้นหาทั้งหมด 7 ผลลัพธ์ ภายในผลลัพธ์ที่ได้ ก็มีการตรวจสอบว่า ข้อมูลหนังสือที่พบที่ชื่อ Harry Potter and the Order of the Phoenix พิมพ์ขึ้นในปี 2002 ใช่หรือไม่ หรือแม้แต่การตรวจสอบชื่อสำนักพิมพ์ที่พิมพ์หนังสือที่ J.K. Rowling แต่งก็ควรจะเป็น Bloomsbury Publishing

2.2 กรณีค้นหาโดยไม่เลือกประเภทการค้นหา

```
[TestMethod]
0 references
public void TestBasicSearchAction5()
{
    InitialController("M_ce51benz", libRepo.Object);
    ViewResult result = controller.Basic("The magic of java", "", 1, 10)
        as ViewResult;
    Assert.IsNull(result.Model as PageList<Book>);
    Assert.AreEqual("Please select search type.",
        result.TempData["ErrorNoti"]);
    Assert.AreEqual("The magic of java", result.TempData["Keyword"]);
}
```

ภายในโค้ดทดสอบกรณีนี้ ก็ใส่ method คล้ายๆกับข้อแรก เพียงแต่พารามิเตอร์ตัวที่สองปล่อยให้เป็น string ว่างๆไป เมื่อเป็นเช่นนั้นแล้วสิ่งที่คาดหวังหลังจากเรียก method นี้คือ มีการเซ็ตค่า TempData[“ErrorNoti”] หรือ output แสดงความผิดพลาดที่จะเอาไปแสดงผลให้กับผู้ใช้ใน Website เป็น Please select search type. อนึ่งในการค้นหานั้นจะเก็บคำค้นหาล่าสุดเอาไว้คืนค่ากลับไปแสดงผลใน input textbox ของหน้าเว็บอีกด้วย ซึ่งโค้ดบรรทัดสุดท้ายจะเป็นการตรวจสอบว่า TempData[“Keyword”] มีการเซ็ตค่าเป็น keyword ตามที่คาดหวังไว้ใช่หรือไม่

2.3 กรณีค้นหาโดยป้อน input ที่ไม่ถูกต้อง

```
[TestMethod]
0 references
public void TestBasicSearchAction10()
{
    InitialController("M_ce51benz", libRepo.Object);
    ViewResult result = controller.Basic("2014a", "Year", 1, 10) as ViewResult;
    Assert.IsNull(result.Model);
    Assert.AreEqual("2014a", result.TempData["Keyword"]);
    Assert.AreEqual("Input string was not in a correct format.",
        result.TempData["ErrorNoti"]);
}
```

ภายในโค้ดทดสอบกรณีนี้ สมมติว่าต้องการค้นหาหนังสือโดยค้นหาตามปีที่พิมพ์ แต่เราส่งคำค้นหาที่ไม่ใช่ตัวเลขไป จะเห็นได้ว่าพารามิเตอร์ตัวแรกที่เป็น keyword ที่ส่งเข้าไปใน method Basic นั้นมีรูปแบบที่ไม่ใช่ตัวเลข (2014a) ผลลัพธ์ที่คาดหวังจากการเรียก method ในกรณีนี้คือ TempData[“ErrorNoti”] มีการเซ็ตค่าแจ้งกลับไปว่า Input string was not in a correct format. และตรวจสอบ TempData[“Keyword”] ในลักษณะที่คล้ายๆกับข้อที่ผ่านมา

2.4 กรณีค้นหาแบบค้นหาโดยข้อมูลที่เกี่ยวข้องทั้งหมดรวมกัน – กรณีปกติ 1

```
[TestMethod]
0 references
public void TestAdvanceSearchAction2()
{
    InitialController("M_ce51benz", libRepo.Object);
    ViewResult result = controller.Advance(new Book
    {
        BookName = "Doraemon",
        Author = "Fujiko F. Fujio",
        CallNumber = "",
        Publisher = "",
        Year = null
    }, 1, 50) as ViewResult;
    Assert.IsNull(result.TempData["ErrorNoti"]);
    Assert.AreEqual(9, (result.Model as PageList<Book>).GetList().Count);
    Assert.AreEqual("Doraemon", result.TempData["BookName"]);
    Assert.AreEqual("Fujiko F. Fujio", result.TempData["Author"]);
    Assert.AreEqual("", result.TempData["Publisher"]);
    Assert.AreEqual(null, result.TempData["Year"]);
    Assert.AreEqual("", result.TempData["Callno"]);
}
```

สำหรับกรณีการทดสอบนี้ได้มีการข้อมูลในการค้นหาคือ

ชื่อหนังสือ: Doraemon

ชื่อผู้แต่ง: Fujiko F. Fujio

หมายเลขเรียกหนังสือ:เว้นว่าง

สำนักพิมพ์:เว้นว่าง

ปีที่พิมพ์:เว้นว่าง

ผลลัพธ์ที่คาดหวังจากการเรียก method Advance ซึ่งเป็น method สำหรับการค้นหาหนังสือขั้นสูงคือ พบหนังสือจำนวน 9 เล่ม ตามข้อมูลหนังสือที่เซ็ตไว้ในตอนแรก พร้อมกับตรวจสอบว่ามีการเก็บข้อมูลคำค้นหาล่าสุดที่ผู้ใช้ได้ทำการค้นหาไว้หรือไม่

2.5 กรณีค้นหาแบบค้นหาโดยข้อมูลที่เกี่ยวข้องทั้งหมดรวมกัน – กรณีปกติ 2

```
[TestMethod]
0 references
public void TestAdvanceSearchAction5()
{
    InitialController("M_ce51benz", libRepo.Object);
    ViewResult result = controller.Advance(new Book
    {
        BookName = "Digital circuit with VHDL",
        Author = "Charoen Vongchumyen",
        CallNumber = "HWA-FL2-2014",
        Publisher = "SE-ED",
        Year = 2009
    }, 1, 10) as ViewResult;
    Assert.IsNull(result.TempData["ErrorNoti"]);
    Assert.AreEqual(1,(result.Model as PageList<Book>).GetList().Count);
    Assert.AreEqual("Digital circuit with VHDL", result.TempData["BookName"]);
    Assert.AreEqual("Charoen Vongchumyen", result.TempData["Author"]);
    Assert.AreEqual("SE-ED", result.TempData["Publisher"]);
    Assert.AreEqual(2009, result.TempData["Year"]);
    Assert.AreEqual("HWA-FL2-2014", result.TempData["Callno"]);
}
```

สำหรับกรณีการทดสอบนี้ได้มีการข้อมูลในการค้นหาคือ

ชื่อหนังสือ: Digital circuit with VHDL

ชื่อผู้แต่ง: Charoen Vongchumyen

หมายเลขเรียกหนังสือ: HWA-FL2-2014

สำนักพิมพ์: SE-ED

ปีที่พิมพ์:2009

กรณีการทดสอบนี้จงใจใส่ข้อมูลที่เฉพาะเจาะจงขึ้น ผลลัพธ์ที่คาดหวังจากการเรียก method Advance ในครั้งนี้คือ พบหนังสือที่มีข้อมูลดังกล่าวเพียงแค่เล่มเดียว พร้อมกับตรวจสอบว่ามีการเก็บข้อมูลคำค้นหาล่าสุดที่ผู้ใช้ได้ทำการค้นหาไว้หรือไม่

2.6 กรณีค้นหาแบบค้นหาโดยข้อมูลที่ไม่พบผลลัพธ์การค้นหาเลย

```
[TestMethod]
0 references
public void TestAdvanceSearchAction3()
{
    InitialController("M_ce51benz", libRepo.Object);
    ViewResult result = controller.Advance(new Book
    {
        BookName = "Computer Organization",
        Author = "Surin",
        CallNumber = "HWA",
        Publisher = "KMITL",
        Year = 2012
    }, 1, 50) as ViewResult;
    Assert.AreEqual("No book result found.", result.TempData["ErrorNoti"]);
    Assert.IsNull((result.Model as PageList<Book>));
    Assert.AreEqual("Computer Organization", result.TempData["BookName"]);
    Assert.AreEqual("Surin", result.TempData["Author"]);
    Assert.AreEqual("KMITL", result.TempData["Publisher"]);
    Assert.AreEqual(2012, result.TempData["Year"]);
    Assert.AreEqual("HWA", result.TempData["Callno"]);
}
```

สำหรับกรณีการค้นหานี้ได้จงใจใส่ข้อมูลการค้นหาที่คาดว่าทำให้ไม่พบผลลัพธ์การค้นหา สำหรับข้อมูลการค้นหาที่ใส่เข้าไปคือ

ชื่อหนังสือ: Computer Organization

ชื่อผู้แต่ง: Surin

หมายเลขเรียกหนังสือ: HWA

สำนักพิมพ์: KMITL

ปีที่พิมพ์:2012

ผลลัพธ์ที่คาดหวังจากการเรียก method นี้คือ TempData[“ErrorNoti”] มีการเซ็ตค่าแจ้ง error กลับไปยังผู้ใช้งานว่า No book result found.

2.7 กรณีค้นหาหนังสือแบบรวดเร็ว กรณีปกติ

```
[TestMethod]
0 references
public void TestQuickSearchAction2()
{
    InitialController("M_ce51benz", libRepo.Object);
    ViewResult result = controller.QuickSearch("Fundamental", 1, 10) as ViewResult;
    Assert.IsNotNull(result.Model as PageList<Book>);
    Assert.AreEqual(1, (result.Model as PageList<Book>).GetPageSize());
    Assert.AreEqual(2, (result.Model as PageList<Book>).GetList().Count);
    Assert.AreEqual("PHY-FL1-8830", (result.Model as PageList<Book>).GetList().
        Where(target => target.BookName ==
            "Fundamental statistics in psychology and education").
            SingleOrDefault().CallNumber);
}
```

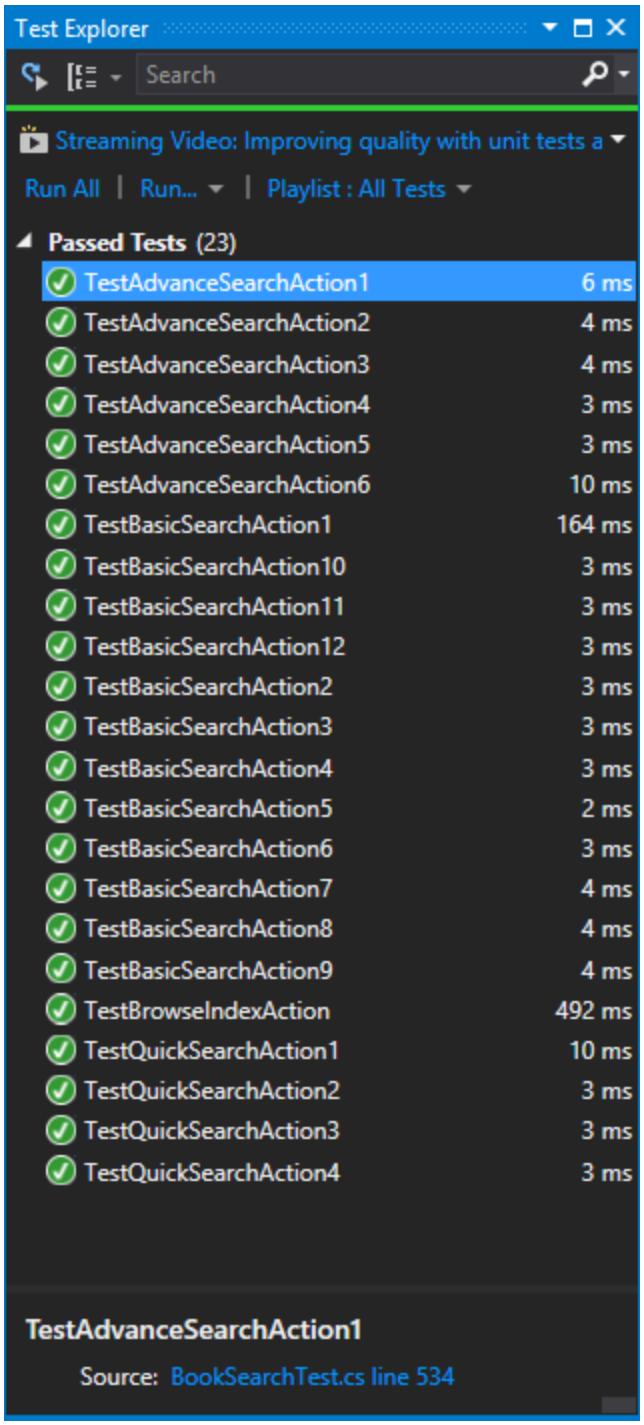
ในกรณีการทดสอบการค้นหาหนังสือแบบรวดเร็วก็ทำการทดสอบในลักษณะที่คล้ายๆ กันกับการค้นหาใน method Basic เพียงแต่ในการค้นหาแบบรวดเร็วนั้นจะค้นหาหนังสือจากชื่อหนังสือเป็นหลัก และไม่มีการเลือกรูปแบบการค้นหา สำหรับกรณีปกติก็คือ พิมพ์คำค้นหาที่คิดว่าน่าจะเจอหนังสือที่ต้องการแล้วค้นหานั้นเอง โดยในกรณีการค้นหานี้ ได้ค้นหาโดยชื่อหนังสือที่มีคำว่า Fundamental อยู่ในชื่อ ผลลัพธ์ที่คาดหวังจากการค้นหาในครั้งนี้คือ พบผลลัพธ์การค้นหาเพียงแค่ 2 รายการ จากข้อมูลหนังสือที่กำหนดไว้ และทำการสุ่มตรวจสอบข้อมูลหมายเลขเรียกหนังสือของผลลัพธ์ที่ได้ว่ามีตรงตามที่คาดเดาไว้หรือไม่

2.8 กรณีค้นหาหนังสือแบบรวดเร็ว กรณีไม่พบผลลัพธ์การค้นหา

```
[TestMethod]
0 references
public void TestQuickSearchAction3()
{
    InitialController("M_ce51benz", libRepo.Object);
    ViewResult result = controller.
        QuickSearch("Faksdsaopdkasdposapdoksapod", 1, 10) as ViewResult;
    Assert.IsNull(result.Model as PageList<Book>);
    Assert.AreEqual("No book result found.", result.TempData["ErrorNoti"]);
}
```

สำหรับการค้นหาในกรณีนี้ เป็นที่แน่นอนว่าจงใจใส่ input ชื่อหนังสือที่มั่วๆไป ผลที่คาดหวังจากการค้นหาคือ TempData[“ErrorNoti”] มีการเซ็ตข้อความ error แจ้งกลับไปยังผู้ใช้เป็น No book result found.

ยังมี test method หรือ test case อีกมากที่ไม่ได้นำเสนอในรายงาน สำหรับผลลัพธ์การทดสอบนั้นผลที่ได้คือ ผ่านทั้งหมด ตามรูปภาพผลลัพธ์การทดสอบในหน้าถัดไป

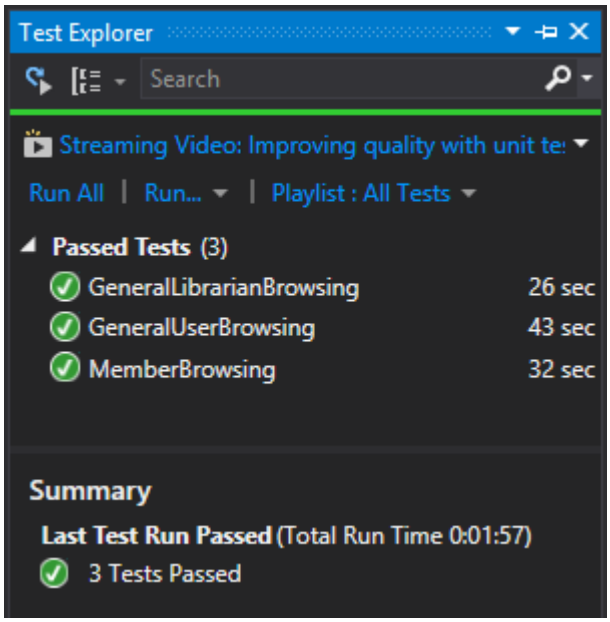


- **Integration test**

1. ทดสอบการทำงานโดยทำการเรียกดู (browse) หน้าทุกหน้าที่เป็นไปได้ของผู้ใช้ในสถานะต่างๆ เพื่อทดสอบว่าผู้ใช้ที่เป็น ผู้ใช้ทั่วไป,สมาชิก,บรรณารักษ์ สามารถเข้าใช้งานได้หรือไม่
   ในการทดสอบนี้นั้น เป็นการทดสอบโดยทำการสั่งเรียกดูหน้าทุกหน้าที่เป็นไปได้ ผ่าน browser ที่สามารถทำงานได้อัตโนมัติไปพร้อมๆกับการตรวจสอบผลลัพธ์ ซึ่งได้มีการเตรียมโค้ดที่สำหรับสั่งให้

browser ทำงานเองอัตโนมัติ โดยสิ่งที่จะตรวจสอบหลักๆ ก็คือ ตรวจสอบว่าผู้ใช้งานทั่วไป สามารถเข้าดูได้เฉพาะหน้าทั่วไปเท่านั้น ถ้าเข้าหน้าที่ถูกจำกัดสิทธิเอาไว้ก็จะเด้งกลับมายังหน้า log in สำหรับสมาชิก สามารถเข้าดูหน้าจัดการบัญชีผู้ใช้งานส่วนบุคคลได้ สามารถดูหน้าทำรายการของสมาชิกได้แต่ไม่สามารถเข้าดูหน้าบริหารจัดการของเจ้าหน้าที่บรรณารักษ์ได้ สำหรับบรรณารักษ์จะเข้าดูได้ทุกหน้า ยกเว้นหน้าทำรายการของสมาชิก

ในการทดสอบนี้ มีเพียง 3 test method แต่ภายใน method นั้นจะเป็น method ย่อยๆอีก ซึ่งภายในจะไปเรียกดูหน้าทุกหน้าที่เป็นไปได้ เนื่องจากต้องเรียกใช้งานกับผู้ใช้งานทั้ง 3 ระดับ โค้ดในลักษณะนี้จึงปรากฏ 3 ครั้ง สำหรับผลการทดสอบนั้น ผ่านทั้งหมด โดยผลการทดสอบเป็นไปตามภาพด้านล่าง



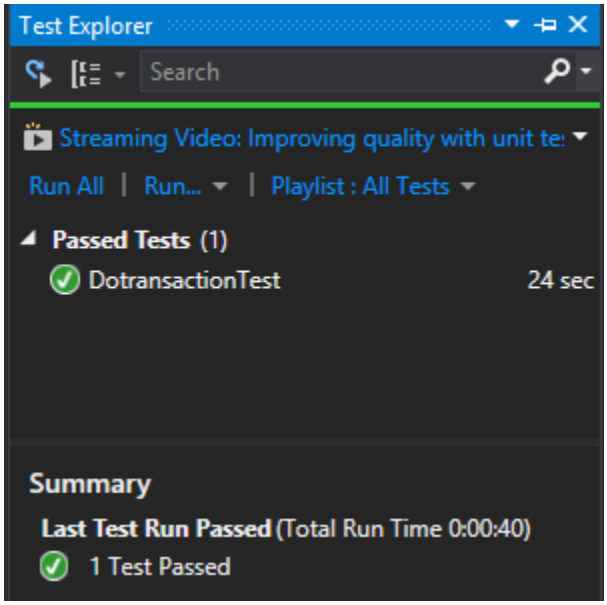
2. ทดสอบการทำรายการยืมหนังสือของสมาชิก 1 คน และทำรายการคืนหนังสือให้กับสมาชิกต่ออีก 1 คนที่ไม่ใช่สมาชิกคนเดิม

ในการทดสอบนี้ยังคงเป็น integration test เหมือนเดิม มีการเตรียมโค้ดที่สำหรับสั่งให้ browser ทำงานเองอัตโนมัติ จุดประสงค์ในการทดสอบนี้คือ ต้องการดูว่า เมื่อทำรายการยืม และคืนหนังสือสำหรับสมาชิกตามที่ต้องการแล้ว ระบบสามารถทำได้สำเร็จตามที่คาดเดาไว้หรือไม่ โค้ดแสดงข้างล่างแสดงส่วนที่ทำการตรวจสอบว่าหลังจากกดทำรายการแล้ว ระบบสามารถทำได้สำเร็จหรือไม่ โดยดูจากข้อความที่ระบบแสดงออกมาใน browser (ใน method AreEqual)

```
app.Browser.FindElement(By.Name("UserID")).SendKeys("23");
app.Browser.FindElement(By.CssSelector("input[type=\"submit\"][value=\"Check\"]")).Submit();
app.Browser.FindElement(By.Name("BookID")).SendKeys("35");
app.Browser.FindElement(By.CssSelector("input[type=\"submit\"][value=\"Submit\"]")).Submit();
Assert.AreEqual(true, app.Browser.FindElement(By.ClassName("noti-green")).
    Text.Contains("Borrow for member"));
```

```
IEnumerator<IWebElement> it = returnbtt.GetEnumerator();
it.MoveNext();
it.Current.Click();
Assert.AreEqual(true, app.Browser.FindElement(By.ClassName("noti-green")).
    Text.Contains("Return successfully."));
app.NavigateTo<AuthenticateController>(c => c.Logout());
app.Route.ShouldMapTo<HomeController>(c => c.Index());
```

ผลการทดสอบคือผ่าน ดังภาพแสดงผลลัพธ์ด้านล่าง



- Directory ที่เก็บไฟล์ code ทดสอบ
    - Unit test เก็บไว้ใน folder LibraryUnitTest
    - Integration test เก็บไว้ใน folder LibraryIntegrationTest
- Test framework
    - Unit test ใช้ Unit Testing Framework ของโปรแกรม Visual studio โดยตรง
    - Integration test ใช้ SpecsFor Mvc

- **Evaluation**

สำหรับการทดลองที่ใช้ในการวัดว่าระบบนั้นสามารถทำงานได้ตามเป้าหมายต่างๆที่ตั้งไว้หรือไม่ ทางกลุ่มได้เลือก 2 การทดลองเพื่อใช้ในการวัด ดังนี้

1.การทดลองวัดความทนทานของระบบ(Robustness)

จุดประสงค์การทดลอง

เพื่อทดสอบการทำงานของเครื่อง server side ที่ใช้ในการทำงาน

สิ่งที่จะวัด

อัตราการ down ของ server

สิ่งที่ต้องใช้ในการทดลอง

Trivial application สำหรับส่ง request ไปหา server

วิธีทำการทดลอง

(a) เริ่มโปรแกรม กำหนดหน้าเว็บไซต์ที่ต้องการเรียกและ ตั้งค่า timeout

(b) ทำการส่ง request หน้าเว็บไซต์ที่ต้องการไปหา server ที่ทดสอบ

(c) รอจนกว่า server จะ response กลับมา ถ้า server ส่ง response กลับมาทัน ให้นับจำนวนครั้งที่สำเร็จ พร้อมกับบันทึกที่ได้ลงใน text file

(d) ถ้า server ไม่ส่ง response กลับมาในเวลาที่กำหนด ให้นับจำนวนครั้งที่ timeout(fail) พร้อมกับบันทึกผลที่ได้ลงใน text file

(e) ทำซ้ำข้อ (b) – (d) ไปเรื่อยๆ เป็นจำนวน 1000 ครั้ง

(f) คำนวณอัตราการ down ของ server แล้วบันทึกผล

ผลที่ได้จากการทดลอง

Timeout ที่ใช้ในการทดลองคือ 500 milliseconds ทำการเรียกหน้า homepage ของ website

ครั้งที่ทำการทดลอง	จำนวนครั้งที่ล้มเหลว	อัตราการ down
1	3	0.3%
2	3	0.3%
3	16	1.6%
เฉลี่ย	7.33	0.73%

สรุปและสิ่งที่ได้จากการทดลอง

จากการทดลองนั้นพบว่าจำนวนครั้งในการ fail ของแต่ละการทดลองไม่ค่อยต่างกันมากนัก ถึงแม้ว่าในการทดลองครั้งสุดท้ายจะมีจำนวนครั้งที่ fail เยอะมากที่สุด แต่เมื่อนำมาเฉลี่ยกับการทดลองอีก 2 ครั้ง ก็พบว่าอัตราการ down โดยเฉลี่ยนั้นไม่ถึง 1% ทั้งนี้ก็ขึ้นกับความเร็วของอินเตอร์เน็ตที่ใช้การทดลองด้วย ซึ่งผลลัพธ์ที่ได้จากการทดลองสามารถสรุปได้ว่า server ที่ใช้ในการรัน web application มีความทนทาน สามารถทนต่อการรับ request จากผู้ใช้ติดต่อกันหลายๆครั้งได้

2.การทดลองค้นหาหนังสือ

จุดประสงค์การทดลอง

เพื่อทดลองค้นหาหนังสือและวัดประสิทธิภาพในการแสดงผลการค้นหาที่ได้ในแต่ละครั้ง

สิ่งที่จะวัด

เวลาในการค้นหาหนังสือโดยเฉลี่ย และผลการค้นหาว่าพบหรือไม่พบ

สิ่งที่ต้องใช้ในการทดลอง

Integration test สำหรับการค้นหาหนังสือ

วิธีทำการทดลอง

(a) เริ่มการทดสอบ ทำการค้นหาหนังสือในแต่ละกรณีการค้นหาที่กำหนดไว้

(b) กดค้นหาแล้วรอจนกว่าหน้าเว็บจะทำการโหลดผลลัพธ์การค้นหา

(c) เมื่อผลลัพธ์การค้นหาปรากฏแล้วให้ตรวจสอบว่ามีผลลัพธ์การค้นหาหรือไม่ ถ้ามีให้วัดเวลาที่ใช้ในการค้นหา ซึ่งเวลาที่ใช้ในการค้นหาจะปรากฏอยู่บนส่วนหัวตารางของผลลัพธ์การค้นหาภายในเว็บ แต่ถ้าไม่พบ ให้ระบุผลการค้นหาว่าไม่พบ(โดยไม่ต้องบันทึกเวลาที่ใช้ในการค้นหา) จากนั้นบันทึกผลลงในไฟล์

(d) ทำซ้ำข้อ (a) – (c) ไปเรื่อยๆ จนกว่ากรณีการค้นหาที่ตั้งไว้จะหมด

(e) ทำการคำนวณหาเวลาในการค้นหาหนังสือโดยเฉลี่ยและสรุปจำนวนครั้งที่ไม่พบลงในตารางบันทึกผล

ผลที่ได้จากการทดลอง

กรณีการค้นหา	ผลการค้นหา		เวลาที่ใช้ในการค้นหา
	พบ	ไม่พบ	
Call number: COM-FL2-2812		✓	
ชื่อหนังสือ: Computer	✓		0.006 วินาที
ชื่อผู้แต่ง: James	✓		0.005 วินาที
สำนักพิมพ์: shogakukan		✓	
ปีที่พิมพ์: 1995	✓		0.005 วินาที
Call number:NOV ชื่อหนังสือ: Sword Art Online Aincrad ชื่อผู้แต่ง: Reki Kawahara สำนักพิมพ์: Zenshu ปีที่พิมพ์: เว้นว่าง	✓		0.025 วินาที
Call number: PE-FL1-0005 ชื่อหนังสือ: Football training ชื่อผู้แต่ง: Graham Taylor สำนักพิมพ์: Leopard Books ปีที่พิมพ์: 1995	✓		0.005 วินาที
Call number: MAT ชื่อหนังสือ: Theory of computation ชื่อผู้แต่ง: รศ.ดร.เกียรติกูล เจียรนัยธนะกิจ สำนักพิมพ์: เว้นว่าง ปีที่พิมพ์: 2009		✓	
Call number: NOV ชื่อหนังสือ: นิทาน	✓		0.008 วินาที

ชื่อผู้แต่ง: เว้นว่าง สำนักพิมพ์: เว้นว่าง ปีที่พิมพ์: เว้นว่าง			

จำนวนครั้งที่ไม่พบ:3

เวลาในการค้นหาโดยเฉลี่ย = $\frac{0.006+0.005+0.005+0.025+0.005+0.008}{6}$ = 0.009 วินาที

<u>สรุปและสิ่งที่ได้จากการทดลอง</u>

จากการทดลองนั้นพบว่าเมื่อทำการสุ่มคำค้นหาในรูปแบบที่เจาะจงข้อมูลและไม่เจาะจงหรือมั่วข้อมูลที่ค้นหา ก็ทำให้ผลการค้นหาที่ได้มีทั้งพบและไม่พบ สำหรับผลการค้นหาที่ในกรณีที่พบนั้นพบว่าเวลาในการค้นหาโดยเฉลี่ยมีค่าที่น้อยในระดับ milliseconds จึงสามารถสรุปได้ว่า web application ที่พัฒนาขึ้นมานั้นสามารถค้นหาหนังสือได้ในเวลาที่รวดเร็วนั่นเอง

<u>บทสรุป</u>

ระบบบริหารจัดการห้องสมุดที่พัฒนา อาจจะมีบางส่วนที่ออกแบบดีบ้างหรือไม่ดีบ้าง ทั้งส่วนที่เป็น front-end กับส่วนที่เป็น back-end แต่ก็ทำให้ผู้จัดทำและสมาชิกในกลุ่มได้เรียนรู้ถึงหลักการทำงานของงานห้องสมุดมากขึ้น ได้เรียนรู้ภาษาโปรแกรมและ framework ที่ใช้ รวมไปถึงหลักการออกแบบเชิงวัตถุ ยังมีคุณสมบัติอีกมากมายที่อยากพัฒนา ผู้จัดทำและสมาชิกในกลุ่มรู้สึกสนุกไปกับการพัฒนาและการค้นหาข้อสงสัยที่เกิดขึ้นในระหว่างการทำ ในขณะที่พัฒนาตัว web application นั้นได้เกิดแนวคิดใหม่ขึ้นเรื่อยๆ เช่น อยากที่จะเพิ่มคุณสมบัตินี้ หรืออยากจะทำให้ส่วนนั้นแสดงผลเป็นแบบที่ต้องการให้ได้ แต่ก็พยายามจัดการและบริหารอยู่ตลอดเวลาจนทำให้งานสุดท้ายที่ออกมาสำเร็จได้ด้วยดี หลังจากทำโครงงานนี้แล้ว ทำให้ผู้จัดทำและสมาชิกในกลุ่มรู้สึกรักและชอบในเรื่องราวเกี่ยวกับการพัฒนาโปรแกรมมากขึ้น

บรรณานุกรม

http://www.asp.net/mvc/mvc5

http://www.asp.net/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application

http://msdn.microsoft.com/en-us/data/dn314429.aspx

http://specsfor.com/SpecsForMvc/

http://www.lib.kmitl.ac.th/

http://www.torontopubliclibrary.ca/blank-search.jsp

https://uwashington.worldcat.org/account/?page=searchItems

http://www.asp.net/web-api/overview/security/enabling-cross-origin-requests-in-web-api

http://www.codeproject.com/Articles/207797/Learn-MVC-Model-view-controller-Step-by-Step-in-7