# การประมาณเวลารอคอยโดยใช้ตัวแบบแถวคอยพลวัต

# WAITING TIME ESTIMATION USING DYNAMIC QUEUING MODEL


นายภูวิศ อนันตโท

MR. PHUWIT ANANTATHO

นายขจรเกียรติ ศักดาศักดิ์

MR. KAJOHNKIAT SAKDASAK



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

# การประมาณเวลารอคอยโดยใช้ตัวแบบแถวคอยพลวัต

# WAITING TIME ESTIMATION USING DYNAMIC QUEUING MODEL

นายภูวิศ อนันตโท
Mr. Phuwit Anantatho
นายขจรเกียรติ ศักดาศักดิ์
Mr. Kajohnkiat SakdaSak

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตร์บัณฑิต
สาขาวิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2555

# WAITING TIME ESTIMATION USING DYNAMIC QUEUING MODEL

MR. PHUWIT ANANTATHO

MR. KAJOHNKIAT SAKDASAK

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INDUSTRIAL ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2012

**คณะวิศวกรรมศาสตร์**

**สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง**

**ใบรับรองปริญญานิพนธ์**

| | |
|---|---|
| **หัวข้อปริญญานิพนธ์** | การประมาณเวลารอคอยโดยใช้ตัวแบบแถวคอยพลวัต<br>Waiting time estimation using dynamic queuing model |
| **นักศึกษา** | นายภูวิศ อนันตโท รหัสประจำตัว 52010003<br>นายขจรเกียรติ ศักดาศักดิ์ รหัสประจำตัว 52010095 |
| **หลักสูตร** | วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมอุตสาหการ |

**อาจารย์ผู้ควบคุมปริญญานิพนธ์**

_____

**(ดร.อุดม จันทร์จรัสสุข)**

| | |
|---|---|
| **หัวข้อปริญญานิพนธ์** | การประมาณเวลารอคอยโดยใช้ตัวแบบแถวคอยพลวัต |
| **นักศึกษา** | นายภูวิศ อนันตโท |
| | นายขจรเกียรติ ศักดาศักดิ์ |
| **หลักสูตร** | วิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมอุตสาหการ |
| | สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง |
| **ปีการศึกษา** | 2555 |
| **อาจารย์ผู้ควบคุมปริญญานิพนธ์** | ดร.อุดม จันทร์จรัสสุข |

## บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้มีวัตถุประสงค์เพื่อประมาณเวลารอคอยของระบบแถวคอยเพื่อช่วยในการตัดสินใจของลูกค้าและพัฒนาระบบงานบริการของผู้ให้บริการ จากการพัฒนาโปรแกรมประมาณเวลารอคอยโดยตัวแบบแถวคอยพลวัต 4 รูปแบบ ซึ่งโปรแกรมจะคำนวณค่าทางสถิติที่เก็บไว้เป็นข้อมูลร่วมกับข้อมูลล่าสุดในระบบเพื่อประมาณเวลารอคอยของผู้ใช้บริการในปัจจุบัน หลักการทำงานของโปรแกรมพัฒนามาจากทฤษฎีแถวคอยผู้ใช้โปรแกรมสามารถกำหนดรูปแบบและจำนวนหน่วยบริการ เพื่อเป็นข้อมูลในการเลือกสมการที่ใช้ในการคำนวณและสะดวกแก่ผู้รับบริการในการทราบเวลารอคอยโดยประมาณจากการทดสอบโปรแกรม โดยเปรียบเทียบผลการทดลองกับโปรแกรมจำลองสถานการณ์ (Arena) ซึ่งเวลารอคอยที่ได้จากโปรแกรมเมื่อเทียบกับโปรแกรมจำลองสถานการณ์ (Arena) โดยพบว่ามีแนวโน้มของเวลารอคอยอยู่ในทิศทางเดียวกัน จึงสรุปได้ว่าโปรแกรมประมาณเวลารอคอยโดยตัวแบบแถวคอยพลวัตสามารถนำไปใช้งานได้จริง

| | |
|---|---|
| **Thesis Title** | Waiting time estimation using Dynamic queuing model |
| **Student** | Mr. Phuwit Anantatho |
| | Mr. Kajohnkiat SakdaSak |
| **Degree** | Bachelor of Engineering in Industrial Engineering |
| | King Mongkut's Institute of Technology Ladkrabang |
| **Academic year** | 2012 |
| **Thesis Advisor** | Dr. Udom Janjarassuk |


## ABSTRACT

This thesis to indicate the approximate waiting time. To help in the decision of the customer and the service provider's system. Developed by the waiting time of the queuing dynamics of the four types, which are calculated by the statistical information was stored together with the latest data for estimate waiting time for now. Program providers to define services type and service channel. The information in the selection equation is used to calculate and easy to get clients in the approximate wait time. The waiting time of a program when compared with the simulation program (Arena) found a trend in the same direction. It can be concluded that the estimated waiting time of the queuing model dynamics can be practical.

# กิตติกรรมประกาศ

ปริญญานิพนธ์เรื่อง การประมาณเวลารอคอยโดยใช้ตัวแบบแถวคอยพลวัต สามารถสำเร็จลุล่วงได้โดยได้รับการสนับสนุน การได้รับคำปรึกษา รวมทั้งการได้รับกำลังใจต่างๆ ซึ่งส่งผลให้การจัดทำโครงการนี้บรรลุเป้าหมายตามที่ได้วางไว้ กลุ่มผู้จัดทำขอขอบพระคุณทุกท่านที่เกี่ยวข้องที่ส่งผลให้ปริญญานิพนธ์ฉบับนี้เสร็จสมบูรณ์

ดร.อุดม จันทร์จรัสสุข อาจารย์ที่ปรึกษาปริญญานิพนธ์ กลุ่มผู้จัดทำขอขอบพระคุณอย่างยิ่งที่คอยให้คำแนะนำ รวมถึงให้ความช่วยในด้านต่างๆตลอดระยะเวลาที่ผ่านมาจนกระทั่งปริญญานิพนธ์ฉบับนี้สามารถสำเร็จลุล่วง

คณาจารย์ทุกท่านที่ประสิทธิ์ประสาทวิชาความรู้ กลุ่มผู้จัดทำขอขอบพระคุณเป็นอย่างสูง สำหรับความรู้ คำแนะนำ และความช่วยเหลือในด้านต่างๆในการจัดทำปริญญานิพนธ์ฉบับนี้

ขอขอบพระคุณบิดามารดาของผู้จัดทำที่ให้โอกาสในการได้รับการศึกษาที่ดี ให้การสนับสนุนและคำแนะนำในทุกๆด้าน และให้กำลังใจพวกเราตลอดมา

ขอขอบพระคุณเพื่อนๆทุกคนสำหรับความช่วยเหลือและกำลังใจที่มีให้กันเสมอมา จนทำให้ปริญญานิพนธ์เล่มนี้สามารถสำเร็จลุล่วงได้ด้วยดี

นายภูวิศ อนันตโท

นายขจรเกียรติ ศักดาศักดิ์

# สารบัญ

หน้า

# สารบัญ (ต่อ)

หน้า

# สารบัญตาราง

หน้า

# สารบัญรูปภาพ

หน้า

# สารบัญรูปภาพ(ต่อ)

หน้า

# บทที่1

# บทนำ

## 1.1 ความเป็นมาและความสำคัญของปริญญานิพนธ์

ในปัจจุบันการเข้าแถวคอยเป็นเรื่องปกติที่พบเห็นได้ทั่วไปในชีวิตประจำวัน ไม่ว่าเป็นการเข้าแถวคอยของคนที่ไปตรวจรักษาที่โรงพยาบาล การเข้าแถวคอยของคนที่ไปรอซื้อตั๋วภาพยนตร์หรือการเข้าแถวคอยในบริการอื่นๆ ซึ่งแถวคอยที่เกิดขึ้นมีสาเหตุมาจากองค์ประกอบหลายอย่างเช่น อัตราการมารับบริการ อัตราการให้บริการ รูปแบบของแถวคอยและพื้นที่ในการให้บริการ เป็นต้น โดยเฉพาะจำนวนผู้รับบริการ เพราะเมื่อลูกค้ามาถึงจุดบริการแต่ยังไม่ได้รับบริการในทันทีนั้น ย่อมเกิดการรอคอย การรอคอยของลูกค้าเป็นเวลานานนั้น สาเหตุหนึ่งอาจมาจากจำนวนผู้ให้บริการที่มีน้อยเกินไป จึงอาจส่งผลให้ลูกค้าบางคนเปลี่ยนใจออกจากการรอรับบริการหรือลูกค้าที่มารับบริการแต่เห็นว่ามีคนรออยู่ในแถวคอยเป็นจำนวนมากจึงไม่เข้ารับบริการ สิ่งเหล่านี้ก่อให้เกิดความเสียหายทางธุรกิจ เพราะการรอคอยถือเป็นต้นทุนทางธุรกิจอย่างหนึ่ง ไม่ว่าจะเป็นด้านการให้บริการหรือด้านอุตสหกรรมซึ่งต้นทุนนี้จึงเป็นส่วนหนึ่งของรายได้ที่สูญเสียไป ในทางกลับกันถ้าจำนวนผู้ให้บริการมีมากในส่วนของลูกค้าอาจมีความสะดวกรวดเร็วไม่ต้องรอคอยนาน แต่ผู้ประกอบการให้บริการจะต้องเสียค่าใช้จ่ายมากขึ้น อีกทั้งเป็นการใช้ทรัพยากรอย่างไม่มีประสิทธิภาพ จากเหตุผลข้างต้น การประมาณเวลารอคอยของลูกค้าในช่วงเวลาต่างๆได้ ย่อมส่งผลดีต่อตัวผู้ให้และรับบริการในด้านการบริหารจัดการเวลาให้เกิดประโยชน์ต่อตนเองมากที่สุด

การประมาณเวลารอคอยของลูกค้าหรือผู้ใช้บริการนั้น สามารถคำนวณได้จากกำหนดค่าบนตัวแปรต่างๆและแทนค่าในสมการตามทฤษฎีแถวคอย (Queuing theory) ซึ่งการแทนค่าตามทฤษฎีดังกล่าวนั้นอัตราการมาถึงค่าลูกค้าเป็นค่าคงที่(Static) แต่ระบบงานบริการที่เกิดขึ้นจริงนั้นอัตราการมาถึงของลูกค้าย่อมมีการเปลี่ยนแปลงอยู่ตลอดเวลา (Dynamic) ทำให้การประมาณเวลารอคอยของลูกค้าโดยใช้ทฤษฎีแถวคอยไม่สามารถใช้ได้ แม้กระทั่งการจำลองสถานการณ์ผ่านโปรแกรม Arena ก็ไม่สามารถตอบสนองต่อความต้องการของลูกค้าได้ทันทีเพราะต้องเสียเวลาในการกำหนดค่าลงบนแบบจำลองต่างๆของโปรแกรมที่เปลี่ยนแปลงตลอดเวลา เรามองเห็นความสำคัญในจุดนี้เพื่อช่วยเพิ่มประสิทธิภาพในการตัดสินใจของลูกค้าและการปรับปรุงคุณภาพของผู้ให้บริการโดยการพัฒนาโปรแกรมประมาณเวลารอคอยโดยตัวแบบแถวคอยพลวัตขึ้น

ระบบแถวคอยหรือระบบคิวแบบพลวัต (Dynamic System) คือ ระบบที่มีการเปลี่ยนแปลงอัตราการมาของวัตถุหรือลูกค้าตามช่วงเวลาต่างๆอยู่ตลอดเวลา ซึ่งเหตุการณ์ที่เกิดขึ้นจะส่งผลให้ต้องมีการเปลี่ยนแปลงในระบบงานบริการไม่ว่าจะเป็นอัตราการให้บริการ จำนวนช่องบริการตามช่วงเวลาต่างๆ โดยปริญญานิพนธ์ฉบับนี้กำหนดขอบเขตการศึกษาของโปรแกรมครอบคลุมรูปแบบแถวคอย 4 แบบดังนี้



1. ช่องทางเดียว ขั้นตอนเดียว
2. ช่องทางเดียว หลายขั้นตอน

3. หลายช่องทาง ขั้นตอนเดียว

4. หลายช่องทาง หลายขั้นตอน

รูปแบบแถวคอยข้างต้น พบได้ทั่วไปตามงานบริการต่างๆในปัจจุบันเช่น ร้านอาหาร โรงพยาบาลและธนาคาร เป็นต้น ซึ่งทราบดีว่าอัตราการมาถึงของลูกค้าไม่คงที่ จึงเป็นที่น่าสนใจในการนำเสนอในปริญญานิพนธ์ฉบับนี้

## 1.2 วัตถุประสงค์

1. เพื่อพัฒนาโปรแกรมสำหรับการประมาณเวลารอคอย โดยใช้ทฤษฎีแถวคอย

2. เพื่อเปรียบเทียบผลของเวลารอคอยที่ได้ระหว่างโปรแกรมที่พัฒนาขึ้นกับแบบจำลองสถานการณ์

## 1.3 ขอบเขต

1. พิจารณาตัวแบบแถวรอคอยเพื่อคำนวณหาเวลารอคอยทั้งหมด 4 แบบ คือ

1.1 ช่องทางเดียว ขั้นตอนเดียว

1.2 ช่องทางเดียว หลายขั้นตอน

1.3 หลายช่องทาง ขั้นตอนเดียว

1.4 หลายช่องทาง หลายขั้นตอน

2. สมมติฐานที่ใช้กำหนดการใช้งานของโปรแกรมคือ ระบบแถวคอยแบบพลวัตที่มีอัตราการมาถึงของลูกค้าในแต่ละช่วงเวลามีลักษณะเดียวกันแต่มีอัตราการมาไม่เท่ากัน (พลวัต) โดยทุกชุดข้อมูลมีการแจกแจงแบบเอ็กซ์โพเนนเชียล

3. เวลาในการให้บริการลูกค้าของแต่ละรูปแบบงานบริการมีการแจกแจงแบบเอ็กซ์โพเน็นเชียลซึ่งมีค่าเฉลี่ยคงที่ไม่ขึ้นกับอัตราการมาของลูกค้าในแต่ละช่วงเวลา

## 1.4 ประโยชน์ที่คาดว่าจะได้รับ

สามารถนำโปรแกรมที่ได้ไปประยุกต์ใช้ประมาณเวลารอคอยของงานบริการได้จริงกับระบบงานหลากหลายประเภท เช่น ร้านอาหารตามสั่ง โรงพยาบาล ธนาคาร หน่วยงานราชการต่างๆ เป็นต้น เพื่อยกระดับการให้บริการและเพิ่มศักยภาพในการแข่งขันทางธุรกิจ เพื่อตอบสนองความต้องการและความพึงพอใจของลูกค้าได้รวดเร็วยิ่งขึ้น

# บทที่ 2
# ทฤษฎีที่เกี่ยวข้อง

ในการทำปริญญานิพนธ์ฉบับนี้เราได้ศึกษาบทเรียนเพิ่มเติมในเนื้อหาวิชาการวิจัยดำเนินงานในส่วนของ ระบบแถวคอย เพื่อให้ได้ข้อมูลอ้างอิง ทฤษฎีและสมการทางคณิตศาสตร์ ที่สามารถนำมาพัฒนาปรับปรุงโปรแกรมประมาณเวลารอคอยด้วยแถวคอยพลวัตได้อย่างถูกต้องซึ่ง ประกอบด้วยเนื้อหาดังนี้

2.1 ทฤษฎีแถวคอยที่มาและความสำคัญของทฤษฎีแถวคอย

2.2 รูปแบบของระบบแถวคอยและปัจจัยที่กำหนดลักษณะพื้นฐานของแถวคอย

2.3 ข้อมูลในระบบแถวคอยและลักษณะข้อมูลที่ใช้คำนวณหาเวลารอคอย

2.4 ตัวแบบแถวคอยการใช้ตัวแบบแถวคอยเพื่อวิเคราะห์และตัดสินใจให้สอดคล้องและเหมาะสม

## 2.1 ทฤษฎีแถวคอย

ทฤษฎีแถวคอยหรือทฤษฎีคิวอิ้ง (Queuing theory) มีชื่อที่มาจากการวิเคราะห์การรอคอยของผู้ใช้บริการหรือลูกค้า(Customer) ที่มาใช้บริการจากผู้ให้บริการ (Server) โดยมีการตั้งเป็นแบบจำลองทางคณิตศาสตร์เพื่อใช้วิเคราะห์ระบบของการให้บริการ ทำให้ทราบข้อมูลเพื่อนำไปปรับปรุงการให้บริการต่อไปได้ เช่น สามารถคำนวณได้ว่ามีลูกค้าโดยเฉลี่ยกี่คนที่กำลังรอเข้ารับบริการ หรือเวลาที่ลูกค้าต้องคอยเข้ารับบริการจากผู้ให้บริการ เป็นต้น

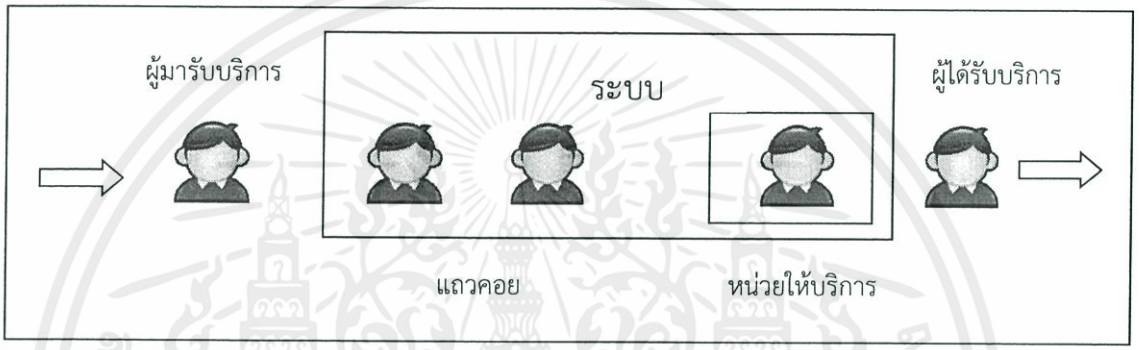
ตัวแบบแถวคอย (Waiting line mode) หรือตัวแบบคิว (Queuing model) นับว่าเป็นตัวแบบที่ใกล้ตัว เนื่องจากในชีวิตประจำวันเราจะเข้าไปเป็นส่วนหนึ่งของระบบตัวคอยอยู่เสมอ เช่น การต่อแถวรอซื้ออาหาร การรอรับยาในโรงพยาบาล การรอรับบริการในธนาคาร ฯลฯ ซึ่งทฤษฎีแถวคอย (Queuing Theory) นี้พัฒนาโดย เอ.เค.เออร์แลง (A.K.Erlang) วิศวกรชาวเดนมาร์ก เมื่อ พ.ศ.2453 เพื่อแก้ปัญหาการรอคอยของรอคอยของผู้ใช้โทรศัพท์ ต่อจากนั้นก็มีผู้ทำการศึกษาระบบแถวคอยลักษณะอื่นๆอย่างกว้างขวาง และพัฒนาตัวแบบแถวคอยขึ้นอีกหลายแบบ ตามลักษณะแถวคอยแบบต่างๆ

อย่างไรก็ตาม ตัวแบบแถวคอยทุกตัวแบบมุ่งเน้นที่จะอธิบายการดำเนินงานของระบบแถวคอยโดยแสดงเป็นข้อมูลตัวเลขต่างๆอย่างชัดเจน จึงทำให้มีการนำตัวแบบแถวคอยไปใช้วิเคราะห์ ระบบแถวคอยหรือช่วยตัดสินใจในการบริการจัดการระบบแถวคอยเพื่อให้มีประสิทธิภาพยิ่งขึ้น

## 2.2 รูปแบบของระบบแถวคอย

การพิจารณาลักษณะพื้นฐานของระบบแถวคอยจะเป็นจุดเริ่มต้นในการนำตัวแบบแถวคอยมาใช้ในการวิเคราะห์การดำเนินงานของระบบ ปัจจัยที่กำหนดลักษณะพื้นฐานของระบบแถวคอยได้แก่ส่วนประกอบที่สำคัญ 3 ส่วน (แสดงดังรูปที่ 2.1) คือ

1. ผู้รับบริการ
2. แถวคอย
3. หน่วยให้บริการ




รูปที่ 2.1 ระบบแถวคอย

### 2.2.1 ลูกค้า

ลักษณะที่สำคัญที่เกี่ยวข้องกับลูกค้า คือ จำนวนประชากรและลักษณะที่ลูกค้าเข้ามารับบริการ ประชากร (Population) คือ ผู้ที่มีโอกาสจะเข้ามาใช้บริการในระบบ ซึ่งระบบแถวคอยบางระบบจะมีผู้ที่สามารถเข้ามาในระบบได้มาก กรณีนี้เรียกว่า ประชากรไม่จำกัด (Infinite) แต่บางกรณีระบบแถวคอยจะมีการกำหนดขอบเขตหรือคุณสมบัติของผู้เข้ามาใช้บริการ ทำให้มีผู้ใช้บริการได้เพียง 20-30 รายเท่านั้น เรียกว่า ประชากรจำกัด (Finite) ในการวิเคราะห์ระบบแถวคอยต่างๆต้องสามารถระบุได้ว่าประชากรของระบบนั้นๆมีจำนวนมากรายหรือน้อยราย

เมื่อกลุ่มประชากรมีความต้องการใช้บริการก็จะเข้ามาสู่ระบบจึงต้องพิจารณาลักษณะการเข้ามารับบริการ (Arrival characteristic) ซึ่งอาจจะเป็นแบบใดแบบหนึ่งดังนี้

1. การเข้ามารับบริการแบบคงที่ (Constant) คือ ลูกค้าเข้ามารับบริการเป็นจำนวนเท่าๆกันในแต่ละช่วงเวลา เช่น วัน ละ 25 คน, ชั่วโมงละ 15 เครื่องหรือวัตถุแต่ละชิ้นมาทุกๆ5 นาทีเป็นต้น ลักษณะเช่นนี้มักพบได้ในโรงงานอุตสาหกรรมที่มีระบบสายการผลิตแบบอัตโนมัติหรือต่อเนื่อง
2. การเข้ามารับบริการแบบสุ่ม (Random) คือ ลูกค้าเข้ามาในลักษณะที่ไม่แน่นอน และการมาถึงหน่วยบริการเป็น อิสระต่อกัน เช่น ผู้ป่วยที่มาโรงพยาบาล ผู้ใช้บริการธนาคาร เป็นต้น

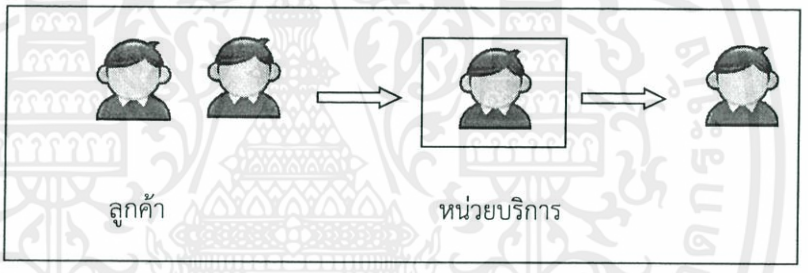
### 2.2.2 แถวคอย

ลักษณะที่สำคัญที่เกี่ยวข้องกับแถวคอย คือ ความยาวของแถวคอยได้แก่ความสามารถในการรองรับลูกค้าของระบบและรูปแบบการจัดระบบแถวคอย

ความยาวของแถวคอยแถวคอยที่เกิดขึ้นเมื่อลูกค้าที่เข้ามารับบริการมีมากกว่าความสามารถในการให้บริการ จึงเกิดการรอคอย ถ้าพื้นที่รอมีน้อยจะทำให้ลูกค้าที่รออยู่ในระบบแถวคอยมีน้อยหรือจำกัดไปด้วย ดังนั้นลูกค้าบางรายจึงอาจไม่สามารถเข้ามาในระบบได้เช่น ร้านอาหาร ร้านทำผม เป็นต้นถ้าแถวคอยเพื่อรับบริการเต็มลูกค้าจะไม่สามารถเข้ามาในระบบได้อีก ต้องตัดสินใจกลับมาในภายหลังหรือใช้บริการที่อื่นแทน อย่างไรก็ตามการจัดการกับแถวที่เหมาะสมจะทำให้ระบบที่มีพื้นที่จำกัดสามารถรับลูกค้าในแถวคอยได้มากขึ้น

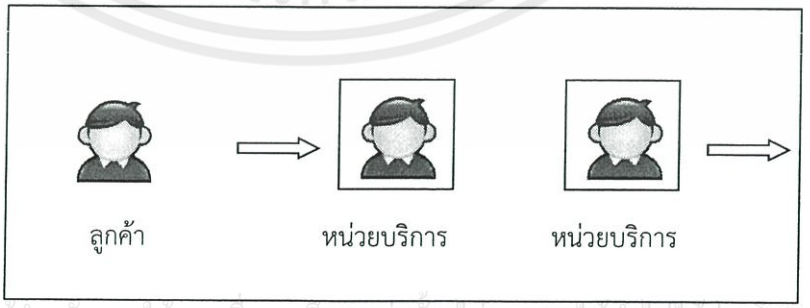
รูปแบบการจัดระบบแถวคอยมีด้วยกันหลายแบบตามลักษณะการให้บริการและจำนวนหน่วยบริการตามทฤษฎี 4 รูปแบบดังต่อไปนี้

1. ระบบแถวคอยแบบช่องทางเดียว-ขั้นตอนเดียว (Single channel- Single server system) คือระบบแถวคอยที่มีหน่วยให้บริการหน่วยเดียว เมื่อลูกค้ารับบริการเสร็จแล้วจะออกไปจากระบบ (แสดงดังรูปที่ 2.2)




รูปที่ 2.2 ระบบแถวคอยแบบช่องทางเดียว–ขั้นตอนเดียว

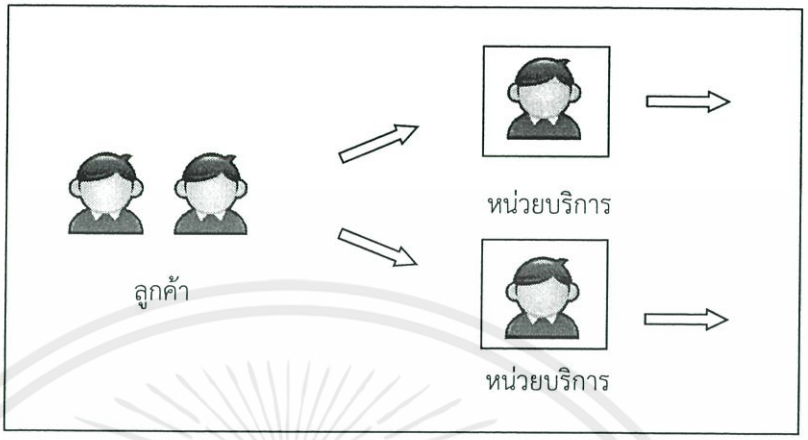
2. ระบบแถวคอยแบบช่องทางเดียว-หลายขั้นตอน (Single channel - Multi servers system) คือ ระบบแถวคอย ที่มีขั้นตอนการให้บริการหลาขั้นตอนซึ่งแต่ละขั้นตอนมีหน่วยให้บริการหน่วยเดียว (แสดงดังรูปที่ 2.3)





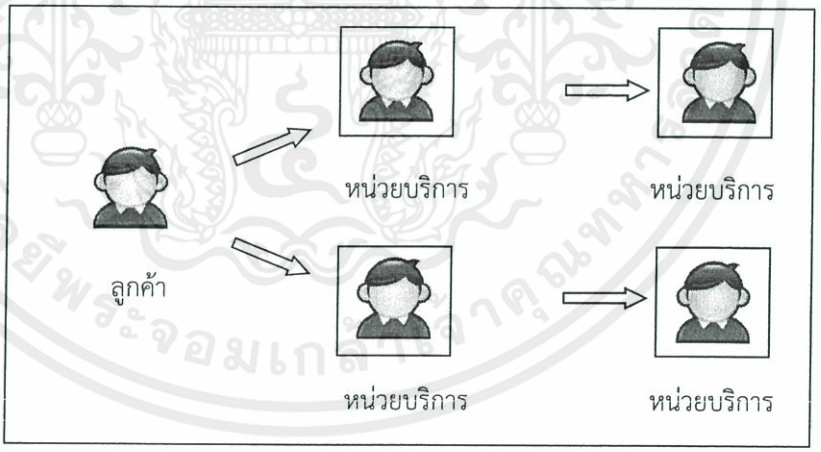

รูปที่ 2.3 ระบบแถวคอยแบบช่องทางเดียว–หลายขั้นตอน

3. ระบบแถวคอยแบบหลายช่องทาง-ขั้นตอนเดียว (Multi channels - Single server system) คือระบบแถวคอยที่มี การให้บริการขั้นตอนเดียว แต่มีหน่วยให้บริการหลายหน่วย (แสดงดังรูปที่ 2.4)




รูปที่ 2.4 ระบบแถวคอยแบบหลายช่องทาง–ขั้นตอนเดียว

4. ระบบแถวคอยแบบหลายช่องทาง-หลายขั้นตอน (Multi channels - Multi servers system) คือ ระบบแถวคอยที่ มีขั้นตอนการให้บริการหลายขั้นตอนและแต่ละขั้นตอนมีหน่วยให้บริการหลายหน่วย (แสดงดังรูปที่ 2.5)




รูปที่ 2.5 ระบบแถวคอยแบบหลายช่องทาง-หลายขั้นตอน

## 2.3 การเก็บข้อมูลในระบบแถวคอย

ข้อมูลที่ใช้ในการคำนวณหาเวลารอคอยจะแบ่งเป็น 2 ลักษณะ ดังนี้

### 2.3.1 ข้อมูลการเข้ารับบริการของลูกค้า

การเข้ารับบริการของลูกค้าแบ่งเป็น 2 ลักษณะดังนี้

1. อัตราการเข้ารับบริการ (Arrival rate) คือ ข้อมูลจำนวนลูกค้าที่เข้ารับบริการในหนึ่งหน่วยเวลาเช่น วินาทีนาที่ชั่วโมง ฯลฯ และทำการบันทึกจำนวนลูกค้าที่มาถึงระบบบริการในแต่ละหน่วยเวลาให้มากพอเพื่อหาค่าเฉลี่ยที่ดีที่สุด โดยปริญญานิพนธ์ฉบับนี้กำหนดให้อัตราการเข้าเราบริการมีการแจกแจงความน่าจะเป็นแบบปัวร์ซอง

2. เวลาเฉลี่ยระหว่างการเข้ารับบริการ (Average interval time) คือ เวลาที่ลูกค้าแต่ละคนมาห่างกันโดยเฉลี่ย เช่น เก็บข้อมูลลูกค้าในช่วงเวลา 3 ชั่วโมงจำนวน 20 คน ดังนั้นระยะเวลาที่ลูกค้ามาห่างกันโดยเฉลี่ยคือ 180/20=9 นาที/คน โดยปริญญานิพนธ์ฉบับนี้กำหนดให้เวลาเฉลี่ยระหว่างการเข้ารับบริการการแจกแจงความน่าจะเป็นแบบเอกซ์โพเนนเชียล

### 2.3.2 ข้อมูลการให้บริการ

การให้บริการแบ่งเป็น 2 ลักษณะดังนี้

1. อัตราการให้บริการ คือ ข้อมูลจำนวนลูกค้าที่ให้บริการในหนึ่งหน่วยเวลาเช่น วินาที นาที ชั่วโมง ฯลฯและทำการบันทึกจำนวนลูกค้าที่ให้บริการได้ในแต่ละหน่วยเวลา เช่น ในเวลา 2 ชั่วโมงบริการได้ 20 คนดังนั้นอัตราการให้บริการเท่ากับ 10 คนต่อชั่งโมง เป็นต้น
2. เวลาในการให้บริการ คือ เวลาที่ใช้ในการให้บริการลูกค้าแต่ละราย เช่น ลูกค้า 300 คนใช้เวลารวม 900 นาที ดังนั้นเวลาในการให้บริการลูกค้าแต่ละคนเฉลี่ยเท่ากับ 900/300= 3 นาที/คน เป็นต้น

## 2.4 ตัวแบบแถวคอย

การเลือกใช้ตัวแบบแถวคอยเพื่อวิเคราะห์และตัดสินใจจำเป็นต้องสอดคล้องและเหมาะสมกับระบบแถวคอยที่เกิดขึ้น ซึ่งรูปแบบแถวคอยที่แตกต่างกันก็จะใช้ตัวแบบที่แตกต่างกันด้วย เพื่อให้สะดวกในการเข้าใจว่าแต่ละตัวแบบควรใช้กับแถวคอยรูปแบบใด นักคณิตศาสตร์ชาวอังกฤษชื่อ ดี.จี. เคนดอล (D.G. Kendall) ได้อธิบายลักษณะตัวแบบแถวคอยให้เข้าใจง่าย เรียกว่า เคนดอลโนเตชั่น (Kendall notation) ดังลักษณะต่อไปนี้

A/B/C/K/N/D

โดย A = การแจกแจงความน่าจะเป็นของการมารับบริการ

B = การแจกแจงความน่าจะเป็นของการรับบริการ

ซึ่งจะใช้อักษรต่างๆแสดงการแจกแจงความน่าจะเป็นของข้อมูลทั้งสองดังต่อไปนี้

M หมายถึงการแจกแจงความน่าจะเป็นแบบปัวส์ซองหรือเอ็กซ์โพเนนเชียล

D หมายถึง ข้อมูลมีลักษณะคงที่

G หมายถึงการแจกแจงความน่าจะเป็นแบบทั่วไป (general distribution)

C = จำนวนหน่วยให้บริการ (channel) โดยใช้ตัวเลขแสดงจำนวนหน่วยให้บริการ

K = ความยาวแถวคอย

N = จำนวนประชากร

ซึ่งจะใช้เครื่องหมาย ∞ แทนในตำแหน่ง K,N ในกรณีที่ไม่จำกัดจำนวนของแถวคอยและประชากรมีจำนวนมากราย

D = ระเบียบการให้บริการ โดยจะใช้ตัวอักษรแสดงระเบียบบริการต่างๆดังต่อไปนี้

First Come First Serve (FCFS) หมายถึง มาก่อนได้รับบริการก่อน

Last Come First Serve (LCFS ) หมายถึง มาทีหลังได้รับบริการก่อน

Service In Random Order (SIRO) หมายถึง ใช้การสุ่มในการให้บริการ

ตัวแบบแถวคอยที่ศึกษาในปริญญานิพนธ์ฉบับนี้มี 2 ตัวแบบ ดังนี้

1. ตัวแบบ M/M/1

ระบบแถวคอยที่ใช้ตัวแบบ M/M/1 จะมีลักษณะดังต่อไปนี้

1. อัตราการเข้ารับบริการเป็นแบบสุ่มและมีการแจกแจงแบบปัวส์ซอง
2. เวลาในการให้บริการเป็นแบบสุ่ม มีการแจงแจกแบบเอ็กซ์โพเน็นเชียล
3. ไม่จำกัดความยาวของแถวคอย
4. จำนวนประชากรมากราย
5. ระเบียบการให้บริการแบบมาก่อนได้รับบริการก่อน (FCFS)
6. เป็นการให้บริการแบบช่องทางเดียวขั้นตอนเดียว

เมื่อพิจารณาจากลักษณะข้างต้นว่าสอดคล้องกับระบบแถวคอยที่ต้องการศึกษาแล้ว ต้องตรวจสอบสมมติฐานของระบบแถวคอยว่าสอดคล้องกันหรือไม่ โดย สมมติฐานของตัวแบบ M/M/1 ได้แก่ อัตราการเข้ารับบริการ ($\lambda$ ) น้อยกว่าอัตราการให้บริการ ($\mu$) หรือ $\lambda < \mu$ ซึ่ง ตัวแบบที่เรานำมาใช้นั้นเป็นตัวแบบของรูปแบบการจัดระบบแถวคอยแบบช่องทางเดียว ขั้นตอนเดียว และแบบช่องทางเดียวหลายขั้นตอนโดยสมการในการคำนวณหาเวลาโดยเฉลี่ยที่ลูกค้าแต่ละคนเสียไปในการรออยู่ในแถวคอย มีดังแสดงในสมการ 2.1

$$W_q = \frac{\lambda}{\mu(\mu-\lambda)} \qquad (2.1)$$

2. ตัวแบบ M/M/s

ระบบแถวคอยที่ใช้ตัวแบบ M/M/s จะมีลักษณะเหมือนตัวแบบ M/M/1 แทบทุกประการเว้นแต่เหมาะสำหรับใช้



กับระบบแถวคอยที่มีขั้นตอนเดียวแต่หลายช่องบริการ ซึ่ง s คือจำนวนช่องบริการ สรุปลักษณะที่ใช้ตัวแบบ M/M/s ดังนี้

1. อัตราการเข้ารับบริการเป็นแบบสุ่มและมีการแจกแจงแบบปัวส์ซอง

2. เวลาในการให้บริการเป็นแบบสุ่ม มีการแจงแจกแบบเอ็กซ์โพเน็นเชียล
3. ไม่จำกัดความยาวของแถวคอย
4. จำนวนประชากรมากราย
5. ระเบียบการให้บริการแบบ มาก่อนได้รับบริการก่อน (FCFS)
6. เป็นการให้บริการแบบหลายช่องทางขั้นตอนเดียว

สมมติฐานของตัวแบบ M/M/s ได้แก่ อัตราการเข้ารับบริการน้อยกว่าผลคูณของจำนวนช่องทางให้บริการกับอัตราการให้บริการหรือ $\lambda < s\mu$

ซึ่งตัวแบบ M/M/s ที่เรานำมาใช้นั้นเป็นตัวแบบของรูปแบบการจัดระบบแถวคอยแบบหลายช่องทาง ขั้นตอนเดียว และแบบหลายช่องทาง หลายขั้นตอนโดยสมการในการคำนวณหาเวลาโดยเฉลี่ยที่ลูกค้าแต่ละคนเสียไปในการรอคอย มีดังนี้

$$p = \frac{\lambda}{s\mu} \tag{2.2}$$

$$P_0 = \frac{1}{\sum_{n=0}^{s-1}\frac{\left(\frac{\lambda}{\mu}\right)^n}{n!} + \left[\frac{\left(\frac{\lambda}{\mu}\right)^s}{s!}\cdot\frac{(s\mu)}{(s\mu-\lambda)}\right]} \tag{2.3}$$

$$L_q = \frac{\left(\frac{\lambda}{\mu}\right)^s \rho}{s!(1-\rho)^2} \tag{2.4}$$

$$W_q = \frac{\lambda}{\mu(\mu-\lambda)} \tag{2.5}$$

ในกรณีที่รูปแบบการให้บริการต่างๆมีอัตราการมาของลูกค้ามากกว่าอัตราให้บริการจะมีวิธีการคำนวณที่แตกต่างกัน โดยมีวิธีการคำนวณเวลารอคอยดังนี้

เวลารอคอยของลูกค้าคนปัจจุบัน = นำเวลาที่ลูกค้าคนก่อนหน้าเข้ารับบริการ + เวลาในการให้บริการโดยเฉลี่ยที่คำนวณจากโปรแกรมคำนวณ –เวลามาของลูกค้าคนปัจจุบัน

สัญลักษณ์ที่ใช้ในการแสดงข้อมูลต่างๆในตัวแบบแถวคอยมีดังนี้

$\lambda$ = อัตราการเข้ารับบริการ(จำนวนลูกค้าที่เข้ารับบริการในหนึ่งหน่วยเวลา)

$\mu$ = อัตราการให้บริการ (ข้อมูลจำนวนลูกค้าที่ให้บริการในหนึ่งหน่วยเวลา)

$1/\mu$ = เวลาโดยเฉลี่ยที่ใช้ในการบริการลูกค้า 1 ราย

P = ความน่าจะเป็นที่ระบบจะทำงาน

$P_0$ = ความน่าจะเป็นที่ระบบว่าง

$L_q$ = จำนวนลูกค้าโดยเฉลี่ยที่อยู่ในระบบแถวคอย

$W_q$ = เวลาโดยเฉลี่ยที่ลูกค้าแต่ละคนเสียไปในการรออยู่ในแถวคอย

โดย ค่า $\lambda$ และ $\mu$ ที่ใช้ในการคำนวณจะต้องเป็นข้อมูลในหน่วยเวลาเดียวกัน

ซึ่งในปริญญานิพนธ์ฉบับนี้เราได้ใช้ทฤษฎีและเนื้อหาดังที่กล่าวมาทั้งหมด เพื่ออ้างอิงและศึกษาการทำงานของระบบแถวคอยและพัฒนาโปรแกรมต่อไป

# บทที่ 3
# การออกแบบและวิธีการดำเนินงาน

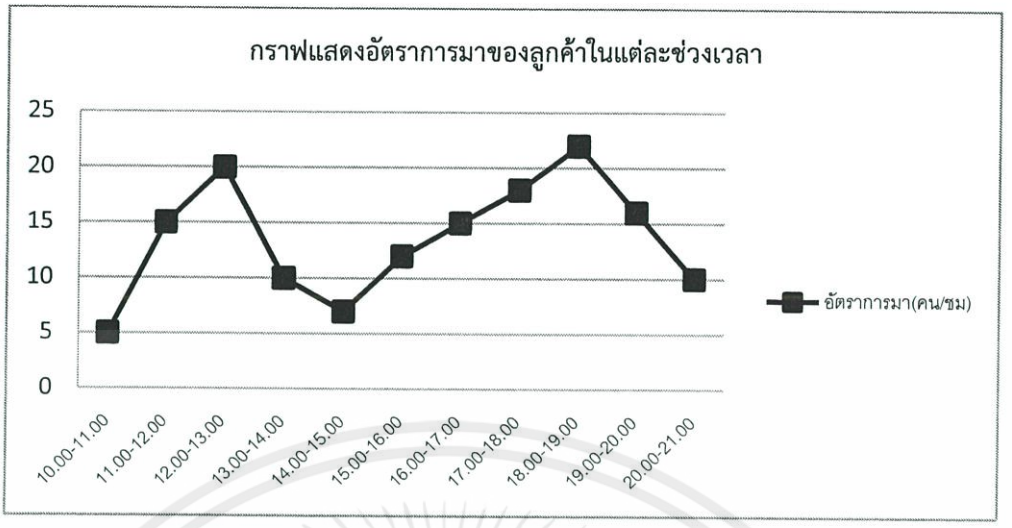
การพัฒนาโปรแกรมประมาณเวลารอคอยโดยตัวแบบแถวคอยพลวัตนั้น นอกจากการศึกษาทฤษฎีที่เกี่ยวข้องแล้ว เราต้องกำหนดแนวคิดและลำดับขั้นตอนการดำเนินงาน เพื่อกำหนดขอบเขตในการทำงาน ซึ่งในบทนี้จะกล่าวถึงแนวคิดของการประมาณเวลารอคอย การสร้างข้อมูลเพื่อใช้เป็นตัวอย่างในการคำนวณหาเวลารอคอย การพัฒนาโปรแกรมที่ใช้สำหรับประมาณเวลารอคอยและการจำลองสถานการณ์ด้วยโปรแกรม Arena

## 3.1 แนวคิดของการประมาณเวลารอคอย

ในงานบริการโดยปกติ รูปแบบการมาของลูกค้าจะมีการแจกแจงแบบเอ็กซ์โพเนนเชียล แต่มีอัตราการมาไม่คงที่ เนื่องจากในงานบริการ เราไม่สามารถทราบเวลาการมาถึงของลูกค้าที่มีความไม่แน่นอนได้ เมื่อผู้ใช้บริการมีมากกว่าที่ระบบสามารถรองรับได้หรือลูกค้าเข้ามาในขณะที่หน่วยบริการกำลังให้บริการผู้อื่นอยู่นั้น การรอรับบริการของลูกค้าที่มาทีหลังจึงเป็นสิ่งที่ไม่สามารถหลีกเลี่ยงได้ โปรแกรมที่เราพัฒนาขึ้นจะมีส่วนช่วยในการตัดสินใจของลูกค้าว่าจะเข้ารับบริการต่อหรือไม่ โดยทำให้ทราบถึงเวลารอคอยโดยประมาณของตนเองในระบบ

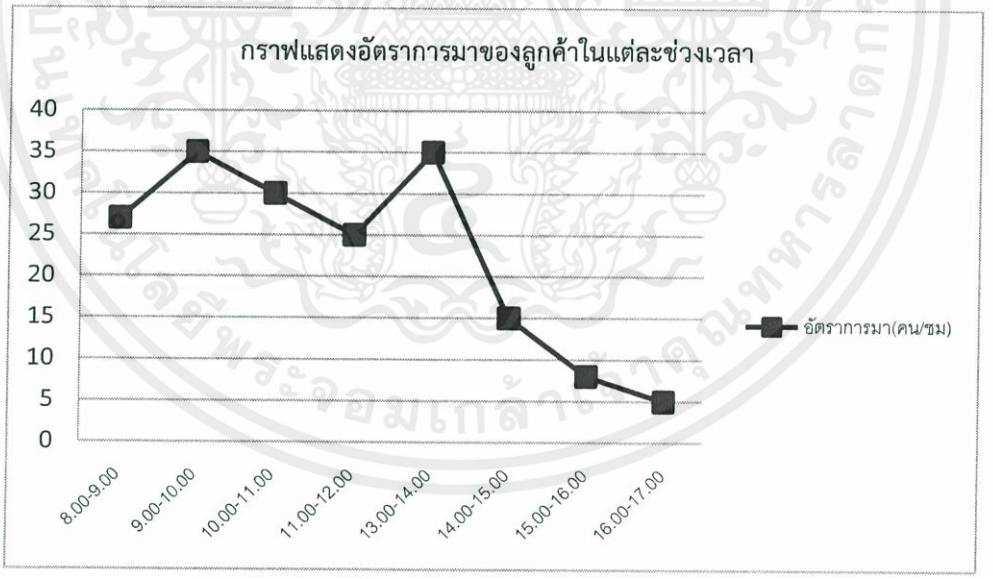
ในงานบริการแต่ละประเภทมักมีอัตราการมาของลูกค้า ($\lambda$) ไม่คงที่และเปลี่ยนแปลงตลอดเวลา ในปริญญานิพนธ์ฉบับนี้เราจึงกำหนดอัตราการมาของลูกค้าที่แตกต่างกันในแต่ละช่วงเวลาด้วยการแจกแจงแบบเอ็กซ์โพเน็นเชียลเท่านั้น และโปรแกรมที่พัฒนาถูกออกแบบให้มีความสามารถในการประมาณเวลารอคอยจากรูปแบบแถวคอยได้ทั้งหมด 4 รูปแบบ เพื่อให้สามารถทดสอบการคำนวณของโปรแกรม เราจึงยกตัวอย่างอัตราการมาของลูกค้าให้มีการเปลี่ยนแปลงในช่วงเวลาต่างๆ ของรูปแบบแถวคอยแต่ละแบบดังนี้ (ในการใช้งานจริง โปรแกรมคำนวณอัตราการมาของลูกค้าจากข้อมูลจริง)

1. Single Channel - Single Server โดยสมมติให้เป็นงานบริการของร้านอาหารตามสั่ง กำหนดเวลาให้บริการตั้งแต่ 10.00 น.–21.00 น. และมีอัตราการมาของลูกค้าในแต่ละช่วงเวลา ซึ่งมีลูกค้าเข้ามาใช้บริการในช่วงเวลาอาหารกลางวันและอาหารเย็น ดังรูปที่ 3.1




รูปที่ 3.1 กราฟแสดงอัตราการมาของลูกค้าในแต่ละช่วงเวลา (Single Channel - Single Server)

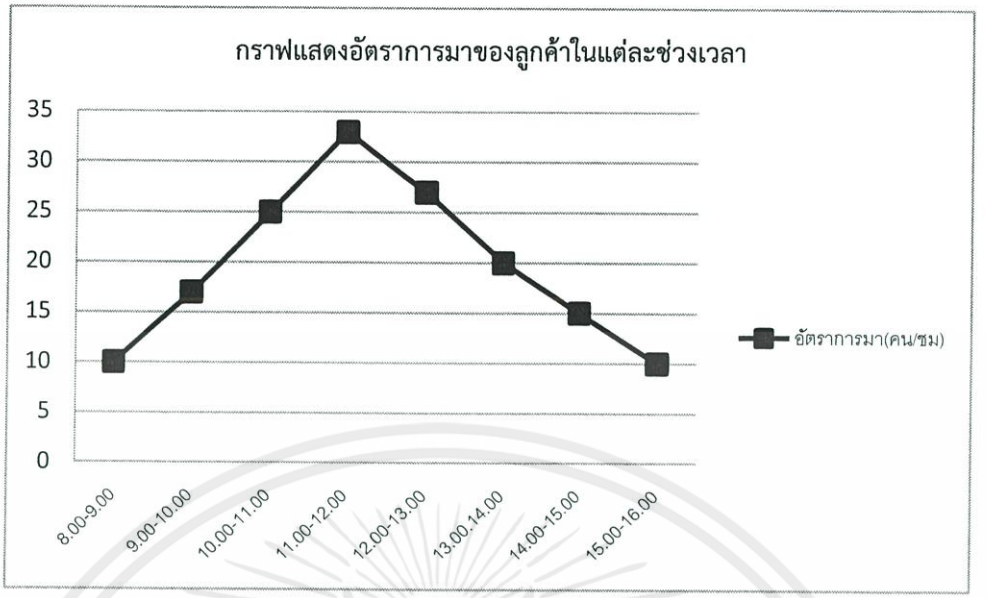
2. Multi Channels - Single Server โดยสมมติให้เป็นงานบริการของแผนกจ่ายยาในโรงพยาบาล กำหนดเวลาให้บริการตั้งแต่ 8.00 น.–17.00 น.และมีอัตราการมาของลูกค้าในแต่ละช่วงเวลา ซึ่งมีลูกค้าเข้ามารับยาตามที่แพทย์สั่งในตลอดทั้งวัน ดังรูปที่ 3.2




รูปที่ 3.2 กราฟแสดงอัตราการมาของลูกค้าในแต่ละช่วงเวลา (Multi Channels - Single Server)

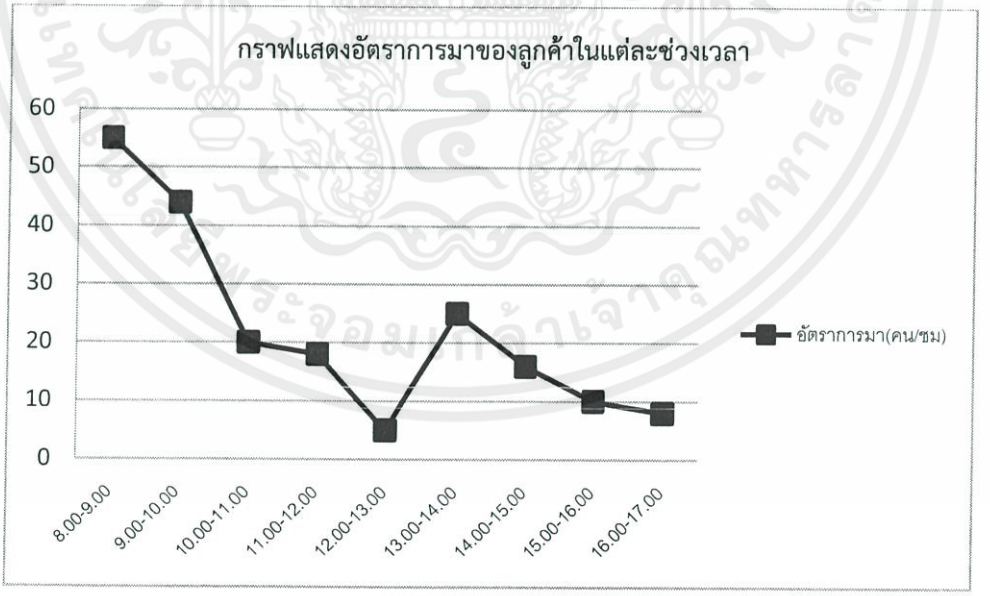


3. Single Channel - Multi Servers โดยสมมติให้เป็นงานบริการของธนาคาร กำหนดเวลา ให้บริการตั้งแต่ 8.00 น. – 16.00 น. และมีอัตราการมาของลูกค้าในแต่ละช่วงเวลา ซึ่งแสดงถึงการใช้บริการของลูกค้าตั้งแต่เวลาเปิดทำการจนสิ้นสุดเวลาทำการ ดังรูปที่ 3.3




รูปที่ 3.3 กราฟแสดงอัตราการมาของลูกค้าในแต่ละช่วงเวลา (Single Channel - Multi Servers)

4. Multi Channels - Multi Servers โดยสมมติให้เป็นงานบริการของแผนกตรวจเลือดในโรงพยาบาล กำหนดเวลาให้บริการตั้งแต่ 8.00 น. – 17.00 น.มีอัตราการมาของลูกค้าในแต่ละช่วงเวลา ซึ่งแสดงจำนวนลูกค้าที่เข้ามาใช้บริการตลอดทั้งวัน ดังรูปที่ 4.4




รูปที่ 3.4 กราฟแสดงอัตราการมาของลูกค้าในแต่ละช่วงเวลา (Multi Channels - Multi Servers)

ข้อมูลเวลาการมาถึงของลูกค้า เวลาในการให้บริการ และเวลาการออกจากระบบของลูกค้าเป็นส่วนสำคัญในการประมาณเวลารอคอยของโปรแกรมที่เราพัฒนาขึ้น เนื่องจากเราจะนำข้อมูลส่วนนี้ไปใช้ในการคำนวณเพื่อให้ได้เวลารอคอยโดยประมาณที่ใกล้เคียงกับความเป็นจริงมากที่สุดและสามารถเปรียบเทียบกับโปรแกรมจำลองสถานการณ์หรือ Arena ได้

## 3.2 การสร้างข้อมูลที่ใช้ในการคำนวณ

ข้อมูลที่ใช้ในการคำนวณคือข้อมูลของเวลาการมาถึงของลูกค้า เวลาในการให้บริการ และเวลาออกจากระบบของลูกค้า โดยข้อมูลถูกสร้างมาจากอัตราการมาของลูกค้าที่กำหนดอยู่ในแต่ละรูปแบบการบริการ และช่วงเวลาของการบริการซึ่งเรากำหนดให้มีความใกล้เคียงกับงานบริการจริงโดยมีขั้นตอนดังนี้

### 3.2.1 การสร้างค่าเวลาการมาของลูกค้า

การสร้างค่าเวลาการมาของลูกค้าที่มีการแจกแจงแบบเอ็กซ์โพเนนเชียลจากค่า $\lambda$ ที่ได้ในหัวข้อ 3.1 โดยการนำอัตราการมาของลูกค้าที่กำหนดไว้ในแต่ละรูปแบบการบริการมาคำนวณด้วยวิธี Inverse ดังแสดงในสมการที่ 3.1

$$T = \frac{-\ln U}{\lambda_i} \tag{3.1}$$

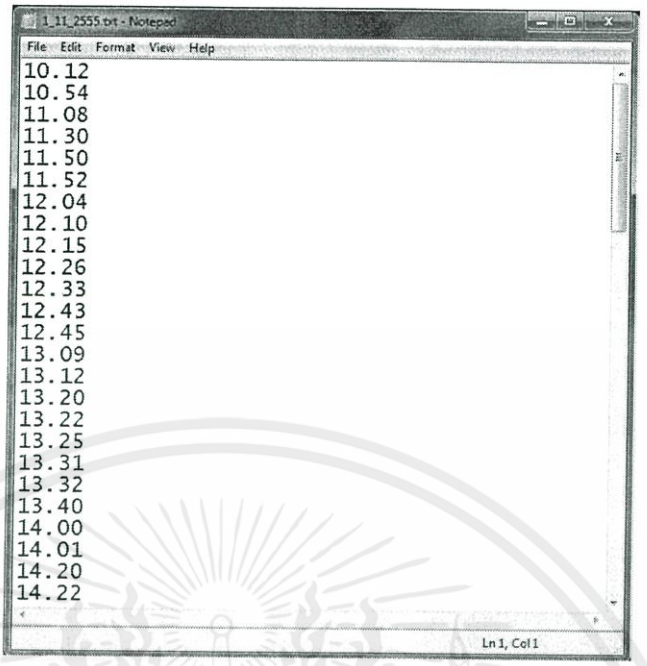
T คือ เวลาที่ลูกค้ามา ณ เวลาหนึ่ง

U คือ ตัวแปรสุ่มที่มีการแจกแจงแบบ Uniform มีค่าระหว่าง 0-1

$\lambda_i$ คือ อัตราการมาของลูกค้าในแต่ละช่วงเวลาที่กำหนดไว้ โดยที่ i = 1,2,3,4,...,N

N คือ จำนวนการแบ่งช่วงเวลาของการให้บริการ

เมื่อสร้างข้อมูลเวลาการมาของลูกค้าแล้ว จากนั้นทำการบันทึกลงในไฟล์ (.txt) ซึ่งจะบันทึกชื่อไฟล์เป็น วันที่_เดือน_ปี เช่น 1_11_2555 เป็นต้น โดยเราจะเก็บข้อมูลของแต่ละรูปแบบงานบริการทั้งหมดรูปแบบละ 30 ไฟล์ ซึ่งสมมติว่าเป็นข้อมูลตั้งแต่วันที่ 1-30 ของเดือน 11 แสดงตัวอย่างดังรูปที่ 3.5




รูปที่ 3.5 ตัวอย่างการบันทึกเวลาการมาของลูกค้าในไฟล์ (.txt)

### 3.2.2 การสร้างข้อมูลเวลาเข้ารับบริการ และเวลาที่ลูกค้าออกจากระบบ

หลังจากที่สร้างข้อมูลเวลาการมาของลูกค้าแล้ว นำข้อมูลที่บันทึกมาคำนวณหาเวลาเข้ารับบริการ และเวลาที่ลูกค้าออกจากระบบ โดยกำหนดเวลาในการให้บริการของแต่ละรูปแบบการบริการเป็นดังตารางดังนี้

ตารางที่ 3.1 ข้อมูลเวลาในการให้บริการเฉลี่ย

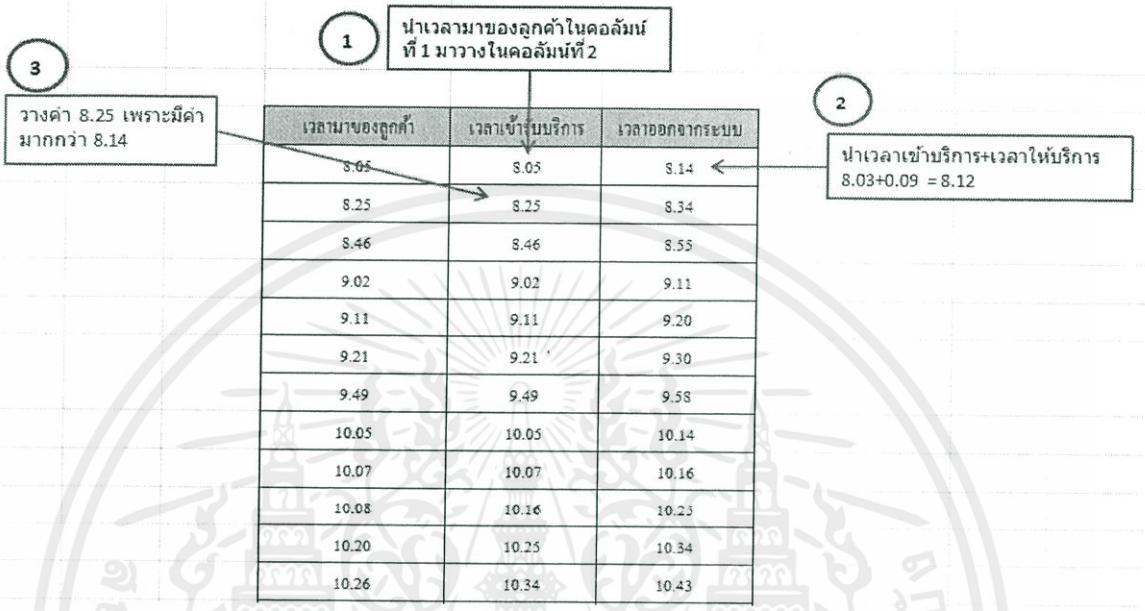
| รูปแบบการให้บริการ | เวลาในการให้บริการเฉลี่ย(นาที) |
|---|---|
| Single Channel Single Server | 5 |
| Multi Channels Single Server | 8 |
| Single Channel Multi Servers | 9 |
| Multi Channels Multi Servers | 9 |

ซึ่งเวลาในการให้บริการเฉลี่ยของแต่ละรูปแบบงานบริการ ประมาณจากเวลาที่เกิดขึ้นจริงในงานบริการนั้นๆ

การสร้างข้อมูลเวลาเข้ารับบริการและเวลาที่ลูกค้าออกจากระบบสามารถสร้างได้จากการนำข้อมูลเวลาการมาของลูกค้าที่ถูกสร้างขึ้นมาคำนวณโดยมีหลักการดังนี้

1. เวลาที่ลูกค้าคนแรกเข้ารับบริการ = เวลาที่ลูกค้าคนแรกเข้ามาในระบบ
2. เวลาที่ลูกค้าคนต่อไปๆเข้ารับบริการ = เวลาที่ลูกค้าคนก่อนหน้าออกจากระบบ (ในกรณีที่เวลามาของลูกค้าคนนั้นๆน้อยกว่าเวลาที่ลูกค้าคนก่อนหน้าออกจากระบบ)

3. เวลาที่ลูกค้าคนต่อไปๆเข้ารับบริการ = เวลาที่ลูกค้าคนนั้นๆเข้ามาในระบบ (ในกรณีที่เวลามาของลูกค้าคนนั้นๆมากกว่าเวลาที่ลูกค้าคนก่อนหน้าออกจากระบบ)
4. เวลาที่ลูกค้าออกจากระบบ = เวลาเข้ารับบริการของลูกค้า+เวลาในการให้บริการโดยเฉลี่ย




| เวลามาของลูกค้า | เวลาเข้ารับบริการ | เวลาออกจากระบบ |
|---|---|---|
| 8.05 | 8.05 | 8.14 |
| 8.25 | 8.25 | 8.34 |
| 8.46 | 8.46 | 8.55 |
| 9.02 | 9.02 | 9.11 |
| 9.11 | 9.11 | 9.20 |
| 9.21 | 9.21 | 9.30 |
| 9.49 | 9.49 | 9.58 |
| 10.05 | 10.05 | 10.14 |
| 10.07 | 10.07 | 10.16 |
| 10.08 | 10.16 | 10.23 |
| 10.20 | 10.25 | 10.34 |
| 10.26 | 10.34 | 10.43 |

รูปที่ 3.6 แสดงวิธีการหาเวลาที่ลูกค้าเข้ารับบริการและออกจากระบบ

เมื่อได้ข้อมูลเวลาที่ลูกค้าเข้าบริการ และเวลาที่ลูกค้าออกจากระบบแล้ว บันทึกข้อมูลทั้งหมดเป็นจำนวน 30 ข้อมูล เพื่อใช้ในการคำนวณต่อไป

## 3.3 การพัฒนาโปรแกรมประมาณเวลารอคอย

โปรแกรมประมาณเวลารอคอยถูกออกแบบให้มีความสามารถในการประมาณเวลารอคอยโดยเฉลี่ยที่ลูกค้าคนหนึ่งๆ ต้องใช้เวลารอในการเข้ารับบริการ โดยการนำข้อมูลเวลาการมาของลูกค้า เวลาที่ลูกค้าเข้ารับบริการ และเวลาที่ลูกค้าออกจากระบบ มาคำนวณโดยใช้สูตรจากทฤษฎีแถวคอย

### 3.3.1 การออกแบบหน้าต่างของโปรแกรม

ขั้นตอนการออกแบบหน้าต่างของโปรแกรมประมาณเวลารอคอยมีขั้นตอนดังนี้

3.3.1.1 <u>กำหนดให้โปรแกรมสามารถเลือกการคำนวณด้วยรูปแบบการบริการ 4 แบบ คือ</u>

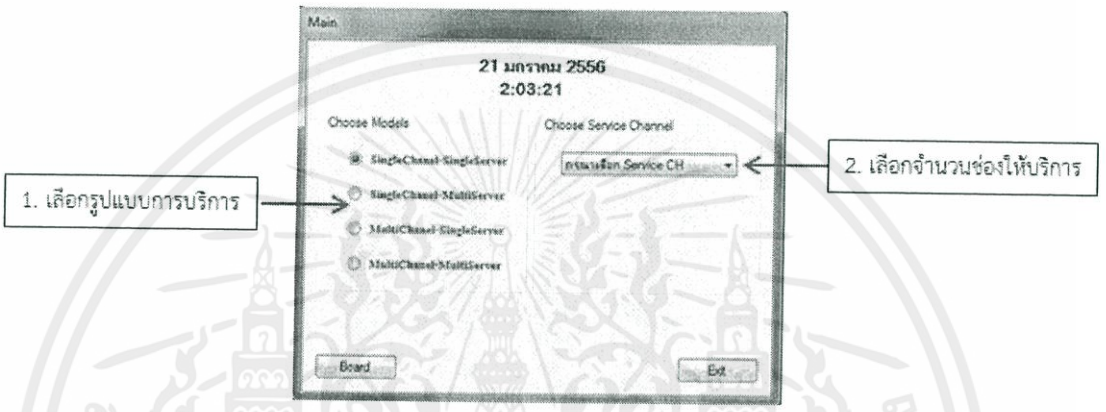
Single Channel Single Serverสามารถเลือก ได้ 1 ช่องบริการ

Multi Channels Single Serverสามารถเลือก ได้ 1 ช่องบริการ

Single Channels Multi Serversสามารถเลือก ได้ 1-3 ช่องบริการ

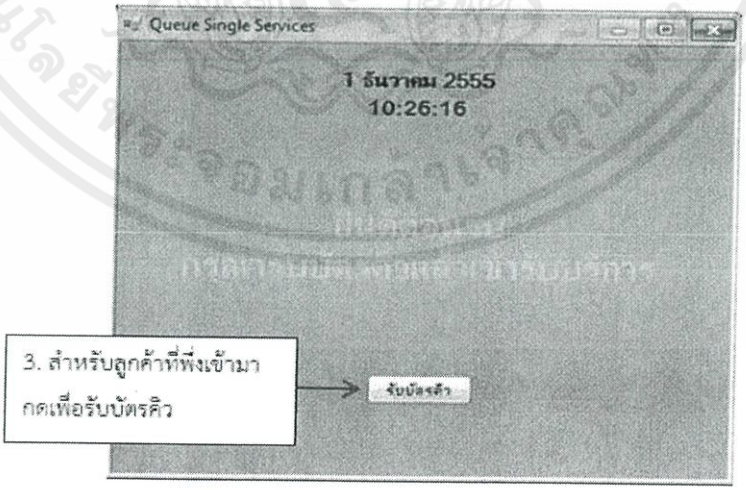
Multi Channels Multi Serversสามารถเลือก ได้ 1-3 ช่องบริการ

3.3.1.2 <u>กำหนดหน้าที่ส่วนประกอบของโปรแกรมไว้ดังนี้</u>



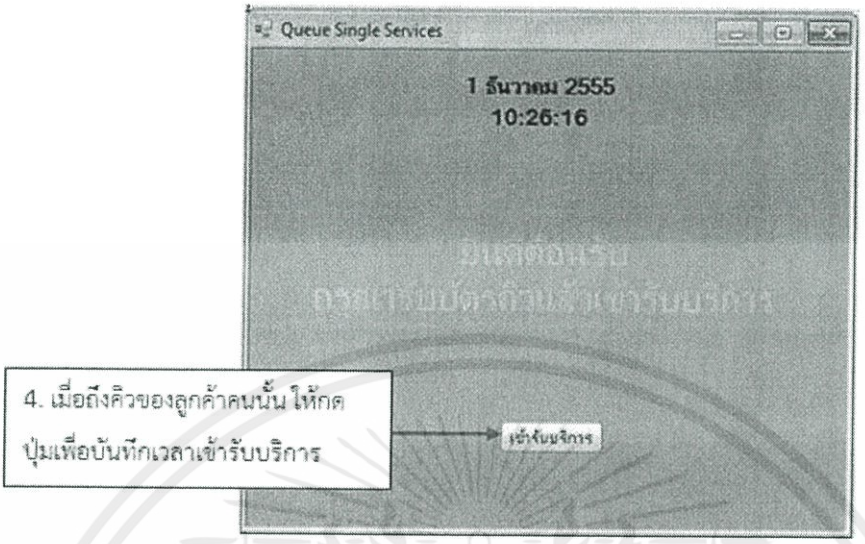

รูปที่ 3.7 เลือกรูปแบบการให้บริการ 4 รูปแบบ

1. ผู้ใช้งานเลือกรูปแบบการให้บริการจากตัวเลือกทั้ง 4 ดังรูปที่ 3.7
2. เลือกจำนวนช่องในการให้บริการ เมื่อเลือกเสร็จแสดงหน้าต่างใหม่ดังรูปที่ 3.8 และรูปที่ 3.9



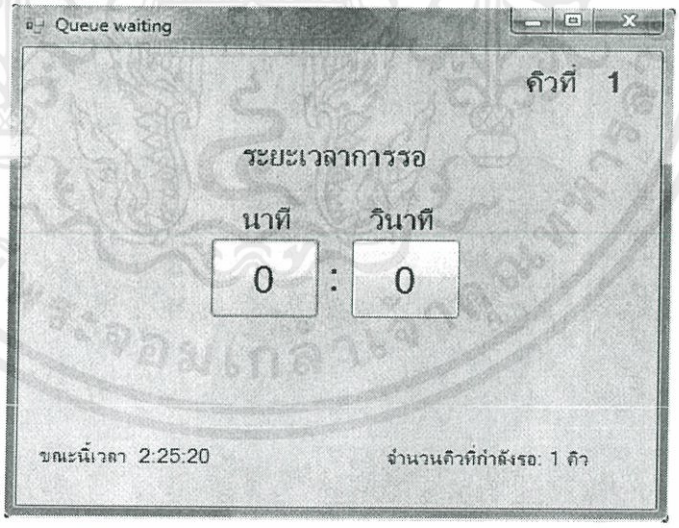



รูปที่ 3.8 แสดงหน้าต่างหลังจากเลือกช่องการให้บริการ



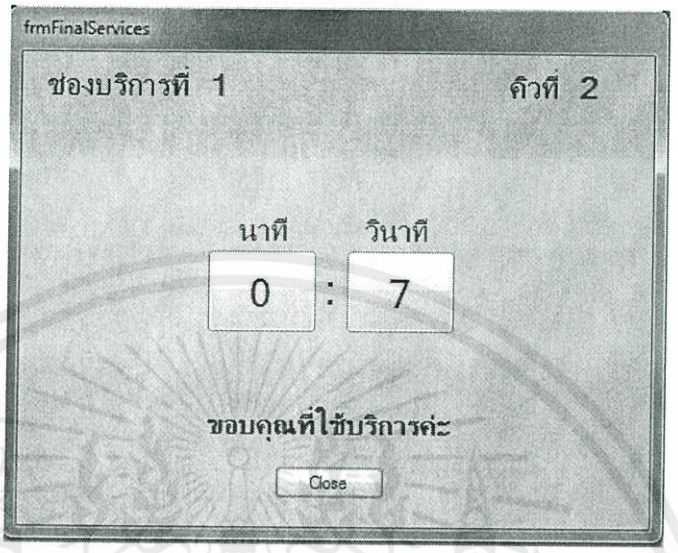

รูปที่ 3.9 แสดงหน้าต่างหลังจากเลือกช่องการให้บริการ

3. สำหรับลูกค้าที่พึ่งเข้ามาจะต้องกดปุ่ม “รับบัตรคิว” เพื่อทราบเวลารอคอย และจำนวนคิวที่ต้องรอ ดังรูปที่ 3.10




รูปที่ 3.10 แสดงหน้าจอบอกเวลารอคอย และจำนวนคิวที่ต้องรอ

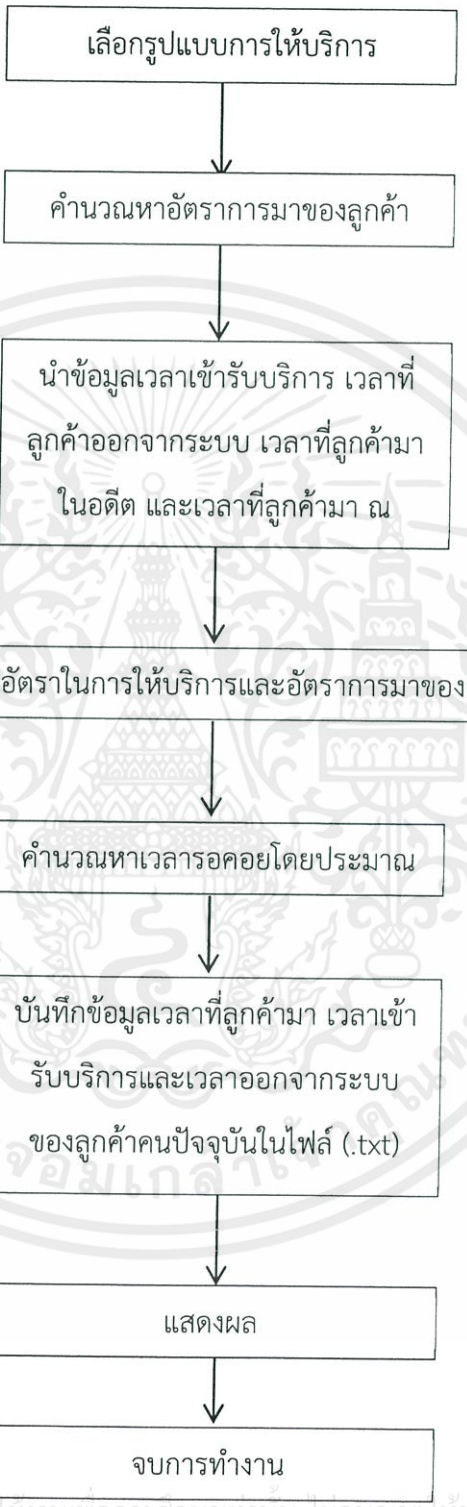
4. กดปุ่ม “เข้ารับบริการ” เมื่อลูกค้าคนหนึ่งๆถึงคิวที่ตนเองเข้ารับบริการเพื่อบันทึกเวลาในการเข้ารับบริการ ดังรูปที่ 3.9 เมื่อกดปุ่ม “เข้ารับบริการ”จะแสดงหน้าต่างบอกเวลาในการให้บริการ ดังรูปที่ 3.11




รูปที่ 3.11 หน้าจอแสดงเวลาในการให้บริการ

### 3.3.2 กำหนดการทำงานของโปรแกรม

ขั้นตอนการทำงานของโปรแกรมประมาณเวลารอคอยมีขั้นตอนดังนี้



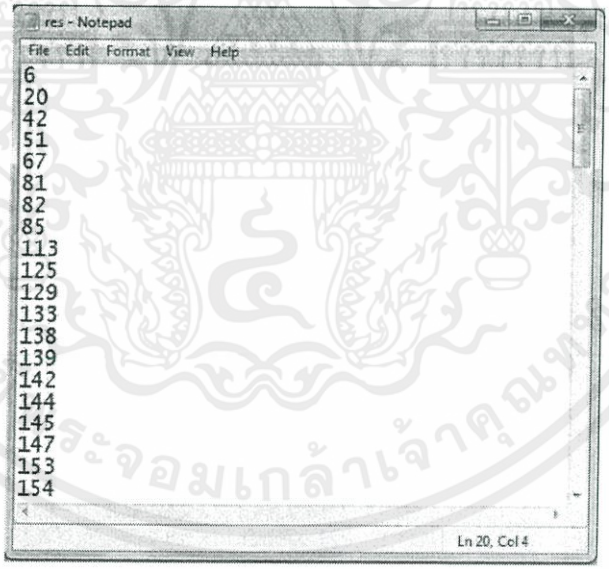



รูปที่ 3.12 Flow Chart การทำงานของโปรแกรมประมาณเวลารอคอย

## 3.4 การจำลองสถานการณ์ด้วยโปรแกรม Arena

การจำลองสถานการณ์ (Simulation) เป็นการรวบรวมวิธีการต่างๆที่ใช้จำลองสถานการณ์จริงหรือพฤติกรรมของระบบ โดยใช้โปรแกรมคอมพิวเตอร์ (Software) เข้ามาช่วย เพื่อศึกษาการไหลของกิจกรรมในรูปแบบต่างๆ ผ่านการเก็บข้อมูล และทำการวิเคราะห์หารูปแบบที่ถูกต้องจากโปรแกรมคอมพิวเตอร์เพื่อปรับปรุงใน อนาคต เนื่องจากในการปฏิบัติงานจริงไม่สามารถที่จะทำการทดลองหรือปรับเปลี่ยนได้ เช่น การขจัดปัญหาที่อยู่นอกเหนือความคาดหมายที่เกิดขึ้น ทำให้กระบวนการผลิตช้าลง ดังนั้นการจำลองสถานการณ์ (Simulation) จะช่วยให้สามารถวิเคราะห์สภาพที่เป็นอยู่ในปัจจุบันของระบบ และช่วยหาแนวทางหรือทางเลือก (Scenario) ที่เหมาะสม ก่อนนำไปใช้กับสถานการณ์หรือการปฏิบัติงานจริง ซึ่งจะช่วยให้ลดความเสี่ยงในการเกิดความผิดพลาด หรือความล้มเหลวได้ นอกจากนี้ยังช่วยให้ประหยัดทั้งค่าใช้จ่าย และเวลาได้อีกทางด้วย

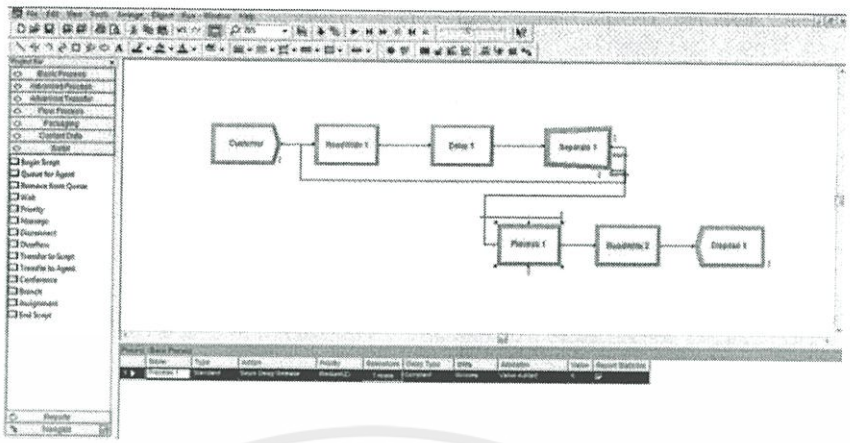
การเปรียบเทียบเวลารอคอยระหว่างโปรแกรมประมาณเวลารอคอยกับการจำลองสถานการณ์นั้น เรากำหนดเวลาการมาถึงของลูกค้าจากข้อมูลที่สร้างด้วยโปรแกรม Microsoft Excel ซึ่งมีการแจกแจงแบบเอ็กซ์โพเน็นเชียล โดยให้เวลาการมาถึงของลูกค้าของโปรแกรมทั้งสองเป็นค่าเดียวกันแสดงตัวอย่างดังรูปที่ 3.15 ซึ่งจะทำให้การเปรียบเทียบค่าเวลารอคอยที่ได้ของลูกค้าแต่ละคนจากโปรแกรมทั้งสองนั้นชัดเจนยิ่งขึ้น เพื่อนำไปใช้วิเคราะห์ความแม่นยำของโปรแกรมที่เราพัฒนาขึ้น โดยมีแบบจำลองสถานการณ์แยกตามลักษณะรูปแบบแถวคอยดังนี้




รูปที่ 3.13 เวลาการมาถึงของลูกค้าของแถวคอยแบบช่องทางเดียว - ขั้นตอนเดียว (นาที )

### 3.4.1 แบบจำลองสถานการณ์ของแถวคอยพลวัตแบบช่องทางเดียว - ขั้นตอนเดียว

การสร้างแบบจำลองสถานการณ์ของแถวคอยพลวัตแบบช่องทางเดียว-ขั้นตอนเดียวนั้นต้องเข้าใจรูปแบบการทำงานระบบแถวคอยคอยก่อน จากนั้นจึงกำหนดโมดูลที่ต้องใช้งานและกำหนดค่าเฉพาะต่างๆลงไปบนโมดูลของแบบจำลองนั้นแสดงดังรูปที่ 3.14



รูปที่ 3.14 ลักษณะแบบจำลองสถานการณ์ของแถวคอยพลวัตแบบช่องทางเดียว-ขั้นตอนเดียว

ซึ่งขั้นตอนการทำงานของแบบจำลองสถานการณ์แถวคอยพลวัตแบบช่องทางเดียว- ขั้นตอนเดียวสามารถอธิบายตามโครงสร้างดังนี้

3.4.1.1 Create Module

คือ โมดูลเริ่มต้นของแบบจำลอง ทำหน้าที่กำหนดระยะเวลาการมาถึงของลูกค้า ในการสร้างแบบจำลองแถวคอยพลวัตแบบช่องทางเดียว-ขั้นตอนเดียว เรากำหนดให้ชื่อ Create Module เป็น Customer และกำหนดค่าให้โมดูลสร้างลูกค้า 1 คนทุกๆ 1 นาที ซึ่งการปล่อยลูกค้าเข้าระบบงานบริการจะถูกกำหนดโดย Separate Module ในลำดับต่อไป

3.4.1.2 Readwrite Module

คือ โมดูลที่ใช้ในการอ่านและบันทึกค่า Text.file โดยในแบบจำลองจะใช้ Readwrite Module2 จุด ได้แก่

1. ใช้ Readwrite Module ต่อจาก Create Module เพื่อดึงค่าเวลาการมาถึง (Read to file) ของลูกค้าตามที่กล่าวไว้ข้างต้นเพื่อกำหนดให้ลูกค้าเข้ามาในระบบตามเวลาที่กำหนดไว้และใช้ Filespreadsheet เลือกตำแหน่งและชื่อไฟล์ของข้อมูลที่บันทึกไว้คือ arrivalsTime\res.txt
2. ใช้ Readwrite Module ต่อจาก Process Module เพื่อบันทึกค่าเวลารอคอยของลูกค้า (Write to file) แต่ละคนโดยกำหนดให้เก็บค่าที่ได้ไว้ที่ Result \res.txt

3.4.1.3 Delay Module

คือ โมดูลที่ใช้ร่วมกับ Readwrite Module เพื่อแสดงเวลาในการทำกิจกรรมของแบบจำลองกำหนดให้เท่ากับ Attribute 1 –TNOW โดย TNOW คือเวลาปัจจุบันของระบบ

3.4.1.4 Separate Module

คือ โมดูลที่ใช้ร่วมกับ Readwrite Module และ Delay Module โดย Separate Module ทำหน้าที่จำแนกและปล่อยลูกค้าให้เข้ามาในระบบตามเวลาที่กำหนด แล้วเริ่มบันทึกเวลาการรอคอยของลูกค้า

3.4.1.5 Process Module

คือ โมดูลที่ใช้กำหนดรูปแบบและเวลาในการให้บริการลูกค้าแต่ละคนโดยแบบจำลองสถานการณ์แถวคอยพลวัตแบบช่องทางเดียว-ขั้นตอนเดียวเรากำหนดให้เป็นงานบริการแบบร้านอาหารตามสั่งมีอัตราการให้บริการคงที่ที่ 5 นาทีต่อคนเพื่อลดความแปรปรวนและสะดวกในการเปรียบเทียบกับโปรแกรมประมาณเวลารอคอยที่พัฒนาขึ้น

3.4.1.6 Dispose Module

Dispose Module คือ โมดูลสุดท้ายที่แสดงจุดสิ้นสุดของแบบจำลองการทำงานของแบบจำลองสถานการณ์เริ่มจาก Create Module กำหนดระยะห่างการมาถึงของลูกค้า 1 คนทุก 1 นาที ต่อด้วย Readwrite Module,Delay Module ,Separate Module ตามลำดับ เพื่อกำหนดเวลาการมาถึงของลูกค้าตามช่วงเวลาที่กำหนดProcess Module จะรับลูกค้าที่ถูกปล่อยมาจากโครงสร้างก่อนหน้าเพื่อเข้าสู่ระบบงานบริการ จากนั้น Readwrite Module ที่ต่อท้ายจะทำหน้าที่บันทึกเวลารอคอยของลูกค้าแต่ละคนในแบบจำลองสถานการณ์ โดยแบบจำลองจะสิ้นสุดที่ Dispose Module

**3.4.2 แบบจำลองสถานการณ์ของแถวคอยพลวัตแบบช่องทางเดียว – หลายขั้นตอน**

ลักษณะแนวคิดและการออกแบบแบบจำลองสถานการณ์ของแถวคอยพลวัตแบบช่องทางเดียว - หลายขั้นตอนนั้นคล้ายกับแบบจำลองสถานการณ์ของแถวคอยพลวัตแบบช่องทางเดียว - ขั้นตอนเดียวตามที่กล่าวไว้ในข้อ 3.4.1 และมีโครงสร้างของแบบจำลองที่เหมือนกัน ดังนั้นในหัวข้อนี้และหัวข้อถัดไปจะกล่าวเพียงจุดที่แตกต่างในแบบจำลองเท่านั้น

3.4.2.1 Readwrite Module

ในแบบจำลองจะใช้ Readwrite Module 2 จุดได้แก่

1. ใช้ Readwrite Module ต่อจาก Create Module เพื่อดึงค่าเวลาการมาถึง (Read to file) ของลูกค้าตามที่กล่าวไว้ข้างต้นเพื่อกำหนดให้ลูกค้าเข้ามาในระบบตามเวลาที่กำหนดไว้และใช้ Filespreadsheet เลือกตำแหน่งและชื่อไฟล์ของข้อมูลที่บันทึกไว้คือ arrivals Time\Drug.txt
2. ใช้ Readwrite Module ต่อจาก Process Module เพื่อบันทึกค่าเวลารอคอยของลูกค้า (Write to file) แต่ละคนโดยกำหนดให้เก็บค่าที่ได้ไว้ที่ Result \Drug Store.txt

3.4.2.2 Process Module

แบบจำลองสถานการณ์แถวคอยพลวัตแบบช่องทางเดียว - หลายขั้นตอนเรากำหนดให้เป็นงานบริการแบบห้องจ่ายยามีอัตราการให้บริการเฉลี่ยคงที่เท่ากับ 8 นาทีต่อคนเพื่อลดความแปรปรวนและสะดวกในการเปรียบเทียบกับโปรแกรมประมาณเวลารอคอยที่พัฒนาขึ้นเช่นเดียวกับที่กล่าวไว้ในหัวข้อที่ 3.4.1.5

### 3.4.3 แบบจำลองสถานการณ์ของแถวคอยพลวัตแบบหลายช่องทาง – ขั้นตอนเดียว

3.4.3.1 Readwrite Module

ในแบบจำลองจะใช้ Readwrite Module 2 จุดได้แก่

1. ใช้ Readwrite Module ต่อจาก Create Moduleเพื่อดึงค่าเวลาการมาถึง (Read to file) ของลูกค้าตามที่กล่าวไว้ข้างต้นเพื่อกำหนดให้ลูกค้าเข้ามาในระบบตามเวลาที่กำหนดไว้และใช้ Filespreadsheet เลือกตำแหน่งและชื่อไฟล์ของข้อมูลที่บันทึกไว้คือ arrivals Time\Bank.txt
2. ใช้ Readwrite Module ต่อจากProcess Module เพื่อบันทึกค่าเวลารอคอยของลูกค้า (Write to file) แต่ละคนโดยกำหนดให้เก็บค่าที่ได้ไว้ที่ Result\Bankk.txt

3.4.3.2 Process Module

แบบจำลองสถานการณ์แถวคอยพลวัตแบบหลายช่องทาง – ขั้นตอนเดียวเรากำหนดให้เป็นงานบริการแบบธนาคารมีอัตราการให้บริการเฉลี่ยคงที่เท่ากับ 9 นาทีต่อคนเพื่อลดความแปรปรวนและสะดวกในการเปรียบเทียบกับโปรแกรมประมาณเวลารอคอยที่พัฒนาขึ้นเช่นเดียวกับที่กล่าวไว้ในหัวข้อที่ 3.4.1.5 แต่กำหนดให้ Capacity ของ Resource เท่ากับ 3 เนื่องจากมีช่องทางในการให้บริการ 3 ช่องทาง

### 3.4.4 แบบจำลองสถานการณ์ของแถวคอยพลวัตแบบหลายช่องทาง – หลายขั้นตอน

3.4.4.1 Readwrite Module

ในแบบจำลองจะใช้ Readwrite Module 2 จุดได้แก่

1. ใช้ Readwrite Module ต่อจาก Create Module เพื่อดึงค่าเวลาการมาถึง (Read to file) ของลูกค้าตามที่กล่าวไว้ข้างต้นเพื่อกำหนดให้ลูกค้าเข้ามาในระบบตามเวลาที่กำหนดไว้และใช้ Filespreadsheet เลือกตำแหน่งและชื่อไฟล์ของข้อมูลที่บันทึกไว้คือ arrivals Time\Blood.txt
2. ใช้ Readwrite Module ต่อจาก Process Module เพื่อบันทึกค่าเวลารอคอยของลูกค้า (Write to file) แต่ละคนโดยกำหนดให้เก็บค่าที่ได้ไว้ที่ Result\Bloodd.txt

3.4.4.2 Process Module

แบบจำลองสถานการณ์แถวคอยพลวัตแบบหลายช่องทาง – หลายขั้นตอน เรากำหนดให้เป็นงานบริการแบบห้องเจาะเลือด มีอัตราการให้บริการเฉลี่ยคงที่เท่ากับ 9 นาทีต่อคนเพื่อลดความแปรปรวนและสะดวกในการเปรียบเทียบกับโปรแกรมประมาณเวลารอคอยที่พัฒนาขึ้นเช่นเดียวกับที่กล่าวไว้ในหัวข้อที่ 3.4.1.5 แต่กำหนดให้ Capacity ของ Resource เท่ากับ 3 เนื่องจากมีช่องทางในการให้บริการ 3 ช่องทางเช่นเดียวกับหัวข้อที่ 3.4.3.2

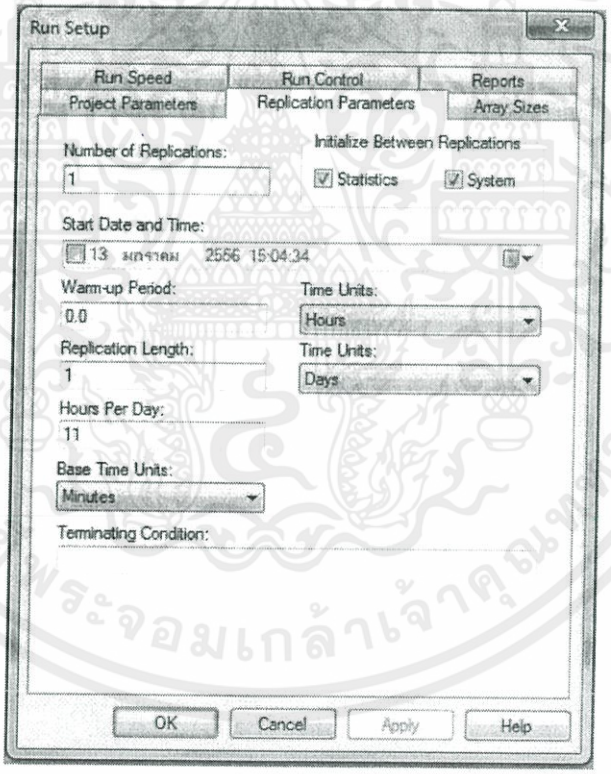
### 3.4.5 การรันผลโปรแกรม

การรันผลโปรแกรมแต่ละแบบจำลองให้ปรับค่าเหมือนกัน ดังนี้

Number of Replications เท่ากับ 1 หมายถึงการรัน 1 รอบเนื่องจากเวลาการมาถึงของลูกค้าเป็นค่าที่กำหนดไว้แล้ว

Replication Length เท่ากับ 1 โดยที่ Time Units เป็น Day หมายถึง ระยะเวลาการรันเท่ากับ 1 วัน

Base Time Units เท่ากับ Minutes




รูปที่ 3.15 การกำหนดค่า Run Setup

ซึ่งส่วนที่แตกต่างกันในแต่ละรูปแบบการให้บริการคือ Hours Per Day โดยมีค่าดังต่อไปนี้

Single Channel - Single Serverกำหนดให้มีค่า 11 หมายถึงระยะเวลาในการให้บริการเท่ากับ 11 ชั่วโมง

Multi Channels - Single Server กำหนดให้มีค่า 9 หมายถึงระยะเวลาในการให้บริการเท่ากับ 9 ชั่วโมง

Single Channel - Multi Servers กำหนดให้มีค่า 8 หมายถึงระยะเวลาในการให้บริการเท่ากับ 8 ชั่วโมง

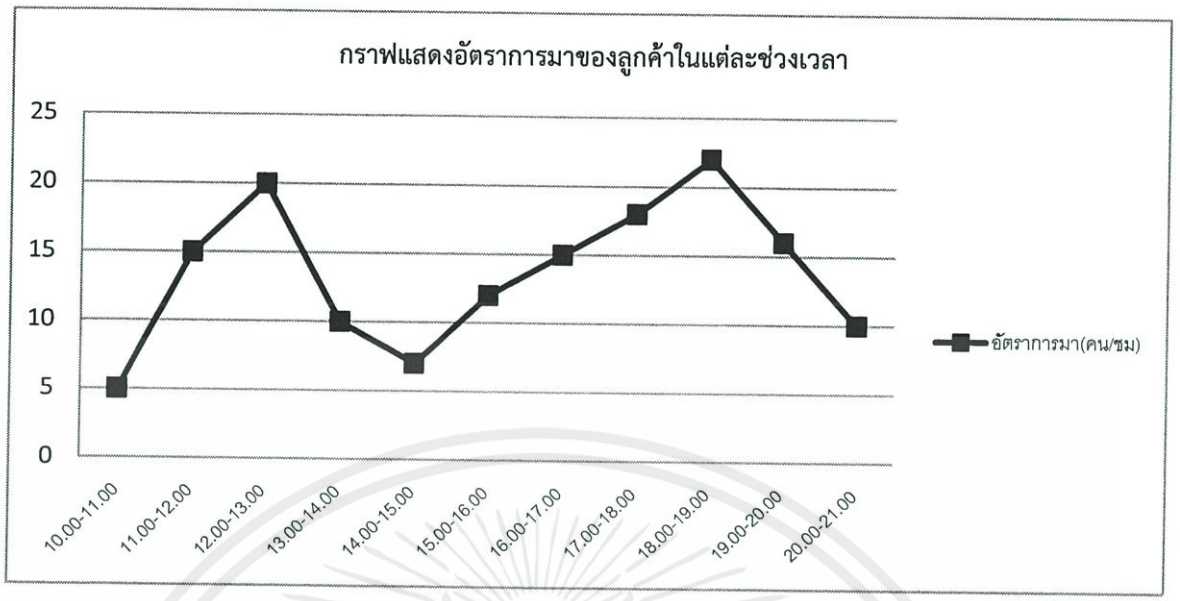
Multi Channels - Multi Servers กำหนดให้มีค่า 9 หมายถึงระยะเวลาในการให้บริการเท่ากับ 9 ชั่วโมง เมื่อกำหนดค่าต่างๆบนโปรแกรม Arena เสร็จแล้วทำการรันโปรแกรมเพื่อให้เกิดการจำลองสถานการณ์และบันทึกเวลารอคอยที่ได้เพื่อนำไปเปรียบเทียบกับผลลัพธ์ที่ได้จากโปรแกรมที่เราพัฒนาขึ้นดังจะกล่าวในบทต่อไป

# บทที่ 4
# ผลการดำเนินงาน

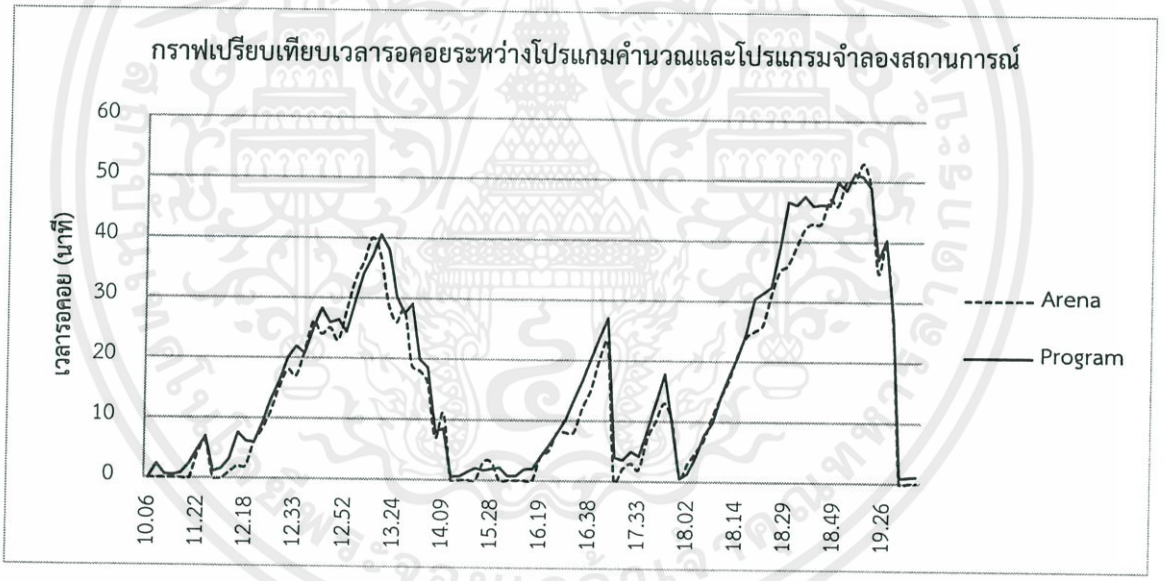
การวัดผลการดำเนินงานของปริญญานิพนธ์ฉบับนี้นั้น เราทำการวัดผลการดำเนินงานโดยการเปรียบเทียบผลลัพธ์ระหว่างเวลารอคอยโดยประมาณที่ได้จากโปรแกรมที่เราพัฒนาขึ้นกับเวลารอคอยโดยเฉลี่ยที่ได้จากโปรแกรมจำลองสถานการณ์ (Arena) โดยกำหนดระยะเวลาการมาของลูกค้าในหนึ่งวันของแต่ละรูปแบบการให้บริการจากการสร้างค่าข้อมูลที่กล่าวในบทที่ 3 เพื่อใช้ในการเปรียบเทียบข้อมูลเวลาการมาของลูกค้าและเวลาในการให้บริการโดยแต่ละรูปแบบการบริการมีดังนี้

## 4.1 กราฟและการวิเคราะห์ผลลัพธ์ที่ได้จากแถวคอยพลวัตแบบช่องทางเดียว–ขั้นตอนเดียว (ร้านอาหารตามสั่ง)

การประมาณเวลารอคอยจากโปรแกรมทั้งสองที่กล่าวข้างต้นนั้น ต้องกำหนดเวลาในการให้บริการลูกค้าแต่ละคนโดยเราจะกำหนดให้เป็นค่าคงที่ค่าเดียวกัน เพื่อลดความแปรปรวนของเวลาในการให้บริการลงทำให้สามารถเปรียบเทียบความความใกล้เคียงกันของการคำนวณระหว่างโปรแกรมคำนวณเวลารอคอยกับโปรแกรมจำลองสถานการณ์ได้ ซึ่งรูปแบบการให้บริการแบบช่องทางเดียว-ขั้นตอนเดียว (ร้านอาหารตามสั่ง) กำหนดไว้ที่ 5 นาทีต่อคนตามที่กำหนดไว้ในหัวข้อที่ 3.2.3 เวลาการมาถึงของลูกค้าและเวลารอคอยโดยประมาณที่ได้แสดงดังตารางที่ ผ 1 นำเวลารอคอยที่ได้จากตารางดังกล่าวมาสร้างแสดงความสัมพันธ์ระหว่างเวลารอคอยกับเวลาการมาถึงของลูกค้า เพื่อเปรียบเทียบแนวโน้มของผลลัพธ์ที่ได้ว่ามีความสัมพันธ์ในทิศทางเดียวกันหรือไม่ แสดงดังรูปที่ 4.2




รูปที่ 4.1 กราฟแสดงอัตราการมาของลูกค้าในแต่ละช่วงเวลา (Single Channel - Single Server)



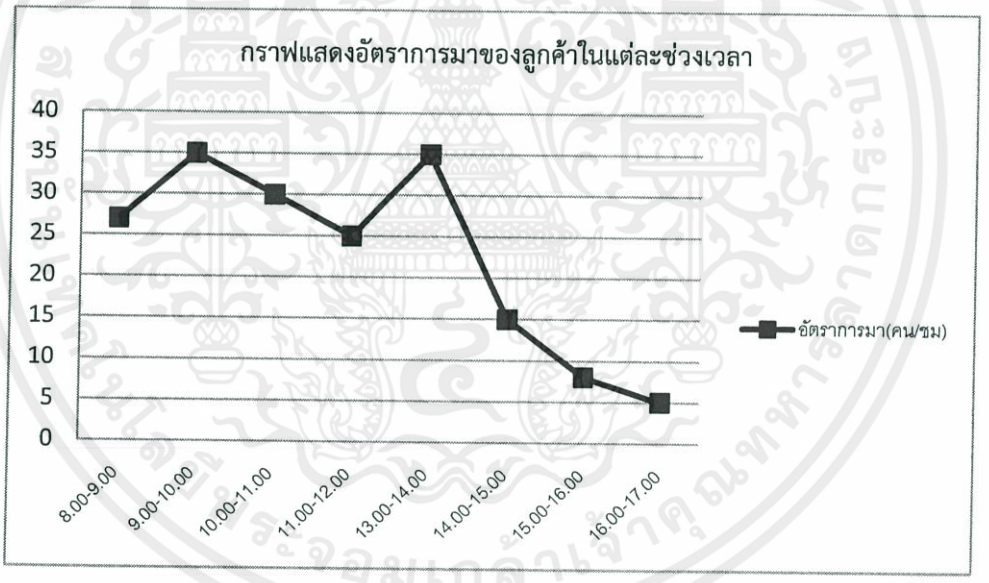

รูปที่ 4.2 กราฟแสดงความสัมพันธ์ระหว่างเวลารอคอยกับเวลาการมาถึงของลูกค้าของแถวคอยพลวัตแบบช่องทางเดียว-ขั้นตอนเดียว (ร้านอาหารตามสั่ง)

จากรูปที่ 4.2 แสดงความสัมพันธ์ระหว่างเวลารอคอยกับเวลาการมาถึงของลูกค้าของแถวคอยพลวัตแบบช่องทางเดียว-ขั้นตอนเดียว (ร้านอาหารตามสั่ง) ค่าเวลารอคอยเฉลี่ยของการบริการทั้งวันจากโปรแกรมจำลองสถานการณ์และโปรแกรมคำนวณเวลารอคอยจากตาราง ผ 1 เท่ากับ 15.914 และ 17.637 นาที ตามลำดับ โดยที่มีความแตกต่างของค่าเฉลี่ยเวลารอคอยทั้งสองเท่ากับ 10.82% จากความสัมพันธ์ดังกล่าวสามารถวิเคราะห์ได้ว่า โปรแกรมทั้งสองที่ใช้ประมาณเวลารอคอยของแถวคอยพลวัตแบบช่องทางเดียว-ขั้นตอนเดียว (ร้านอาหารตามสั่ง) มีความสัมพันธ์กัน

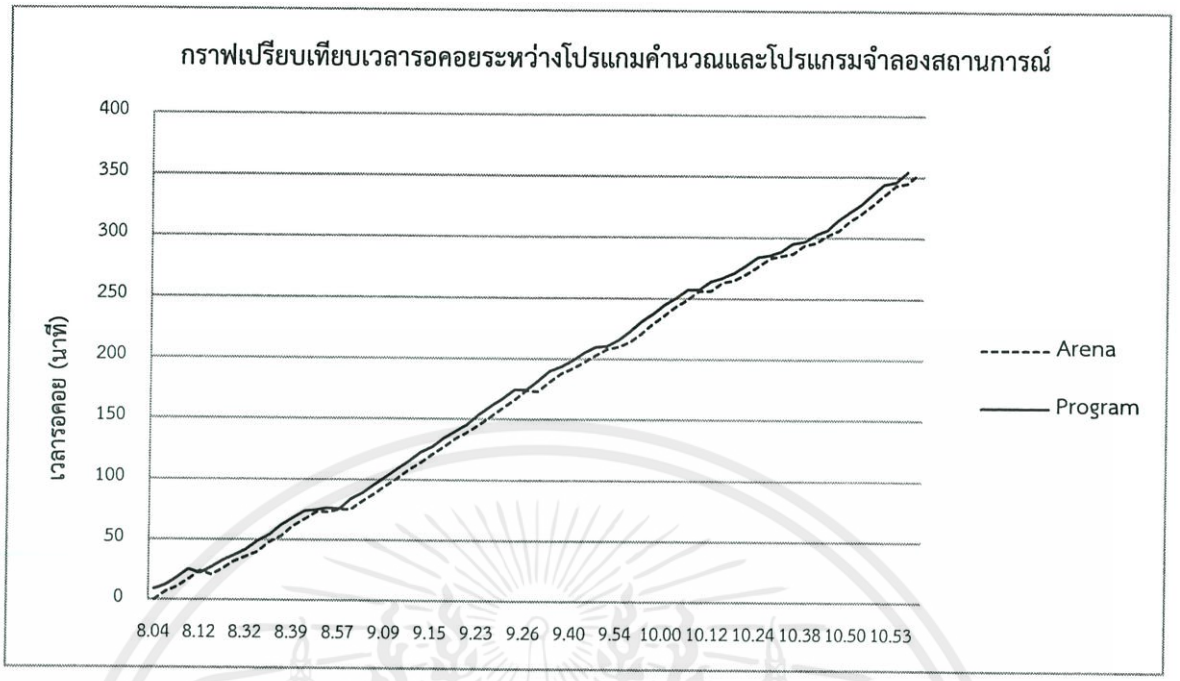
เนื่องจากแนวโน้ม และการช่วงการขึ้น-ลงของเวลารอคอยตามเวลาการถึงของลูกค้าเป็นไปในทิศทางเดียวกัน ค่าเวลารอคอยของลูกค้ามีความใกล้เคียงหรือเท่ากันในบางช่วงเวลาและไม่พบค่าผิดปกติ (พุ่งสูงหรือลดต่ำเกินเมื่อเทียบกับค่าที่ได้จากอีกโปรแกรม) ที่จุดใดๆ เมื่อนำมาเปรียบเทียบกับกราฟแสดงอัตราการมาของลูกค้าในแต่ละช่วงเวลาดังรูปที่ 4.1 จะพบว่ามีลักษณะการขึ้น-ลงของเวลารอคอยที่สัมพันธ์กับจำนวนลูกค้าแต่ละช่วงเวลา

## 4.2 กราฟและการวิเคราะห์ผลลัพธ์ที่ได้จากแถวคอยพลวัตแบบช่องทางเดียว–หลายขั้นตอน (ห้องจ่ายยา)

ดังที่ได้กล่าวในหัวข้อ 4.1 ได้ ซึ่งรูปแบบการให้บริการแบบช่องทางเดียว-หลายขั้นตอน(ห้องจ่ายยา) จะกำหนดเวลาในการให้บริการไว้ที่ 8 นาทีต่อคนตามที่กล่าวไว้ในหัวข้อ 2.3.3 เวลาการมาถึงของลูกค้าและเวลารอคอยโดยประมาณที่ได้แสดงดังตารางที่ ผ 2 นำเวลารอคอยที่ได้จากตารางดังกล่าวมาสร้างกราฟแสดงความสัมพันธ์ระหว่างเวลารอคอยกับเวลาการมาถึงของลูกค้า เพื่อเปรียบเทียบแนวโน้มของผลลัพธ์ที่ได้ว่ามีความสัมพันธ์ในทิศทางเดียวกันหรือไม่ แสดงดังรูปที่ 4.4




รูปที่ 4.3 กราฟแสดงอัตราการมาของลูกค้าในแต่ละช่วงเวลา (Multi Channels - Single Server)



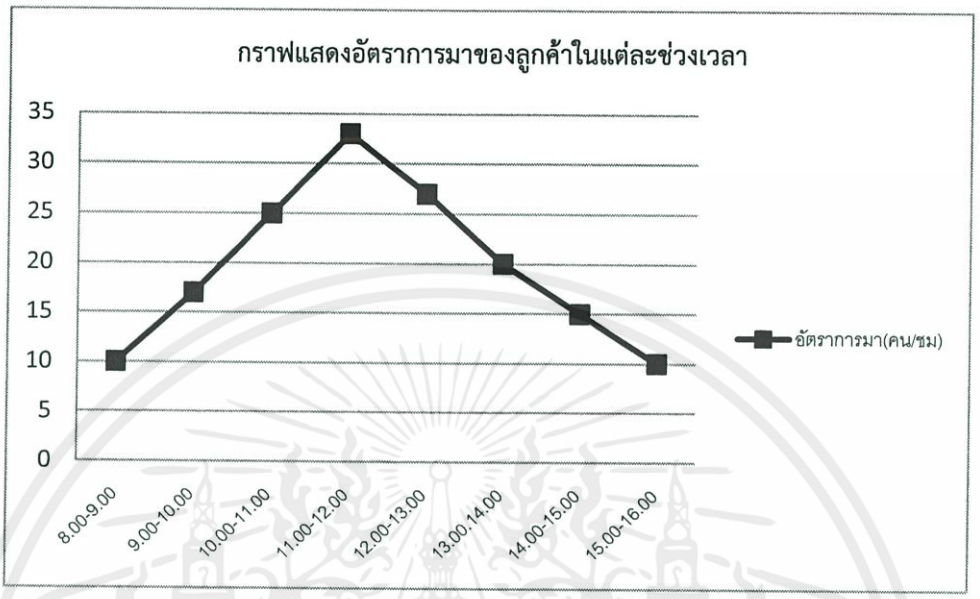

รูปที่ 4.4 กราฟแสดงความสัมพันธ์ระหว่างเวลารอคอยกับเวลาการมาถึงของลูกค้าของแถวคอยพลวัตแบบช่องทางเดียว-หลายขั้นตอน (ห้องจ่ายยา)

จากรูปที่ 4.4 แสดงความสัมพันธ์ระหว่างเวลารอคอยกับเวลาการมาถึงของลูกค้าของแถวคอยพลวัตแบบช่องทางเดียว-หลายขั้นตอน (ห้องจ่ายยา) ค่าเวลารอคอยเฉลี่ยของการบริการทั้งวันจากโปรแกรมจำลองสถานการณ์และโปรแกรมคำนวณเวลารอคอยตาราง ผ 2 เท่ากับ 167.097 และ 176.212 นาที ตามลำดับ โดยที่มีความแตกต่างของค่าเฉลี่ยเวลารอคอยทั้งสองเท่ากับ 5.46% จากความสัมพันธ์ดังกล่าวสามารถวิเคราะห์ได้ว่า โปรแกรมทั้งสองที่ใช้ประมาณเวลารอคอยของแถวคอยพลวัตแบบช่องทางเดียว-หลายขั้นตอน(ห้องจ่ายยา) แม้กราฟที่ได้จากโปรแกรมประมาณเวลารอคอยจะมีค่าเวลารอคอยสูงกว่าค่าที่ได้จากโปรแกรม Arena ทุกจุดซึ่งเกิดจากการประมาณค่าของข้อมูลในอดีตที่โปรแกรมนำมาใช้ในการคำนวณแต่ยังมีความสัมพันธ์กันเนื่องจากแนวโน้มของเวลารอคอยที่เพิ่มสูงขึ้นตามเวลาการถึงของลูกค้าเป็นไปในทิศทางเดียวกัน เนื่องจากระบบไม่สามารถระบายลูกค้าออกจากระบบได้ทันเมื่อลูกค้ามีปริมาณมากขึ้นจึงเกิดการสะสมของเวลารอคอย ซึ่งสอดคล้องกับอัตราการมาถึงของลูกค้าจำนวนมากในช่วงชั่วโมงแรก ดังกราฟแสดงอัตราการมาของลูกค้าในแต่ละช่วงเวลาดังรูปที่ 4.3

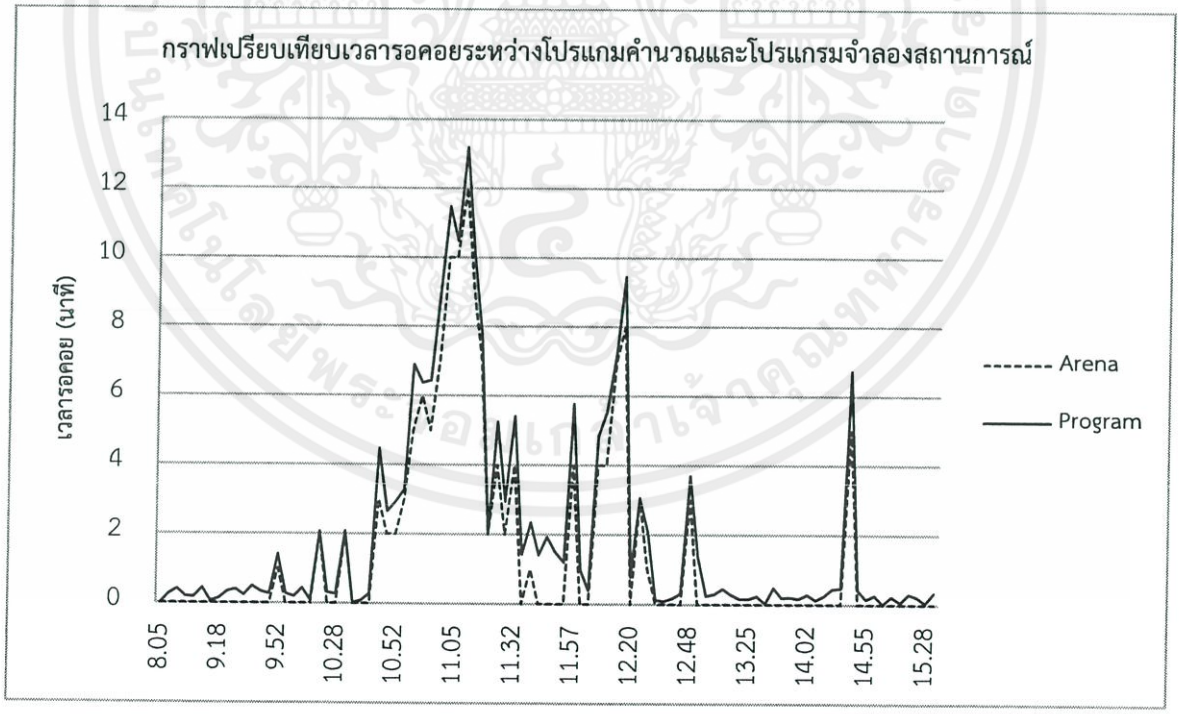
## 4.3 กราฟและการวิเคราะห์ผลลัพธ์ที่ได้จากแถวคอยพลวัตแบบหลายช่องทาง–ขั้นตอนเดียว (ธนาคาร)

ดังที่ได้กล่าวในหัวข้อ 4.1 ได้ ซึ่งรูปแบบการให้บริการแบบหลายช่องทาง-ขั้นตอนเดียว (ธนาคาร) จะกำหนดเวลาในการให้บริการไว้ที่ 9 นาทีต่อคนตามที่กล่าวไว้ในหัวข้อ 2.3.3 เวลาการมาถึงของลูกค้าและเวลารอคอยโดยประมาณที่ได้

แสดงดังตารางที่ ผ 3 นำเวลารอคอยที่ได้จากตารางดังกล่าวมาสร้างกราฟแสดงความสัมพันธ์ระหว่างเวลารอคอยกับเวลาการมาถึงของลูกค้า เพื่อเปรียบเทียบแนวโน้มของผลลัพธ์ที่ได้ว่ามีความสัมพันธ์ในทิศทางเดียวกันหรือไม่ ดังรูปที่ 4.6




รูปที่ 4.5 กราฟแสดงอัตราการมาของลูกค้าในแต่ละช่วงเวลา (Single Channel - Multi Servers)

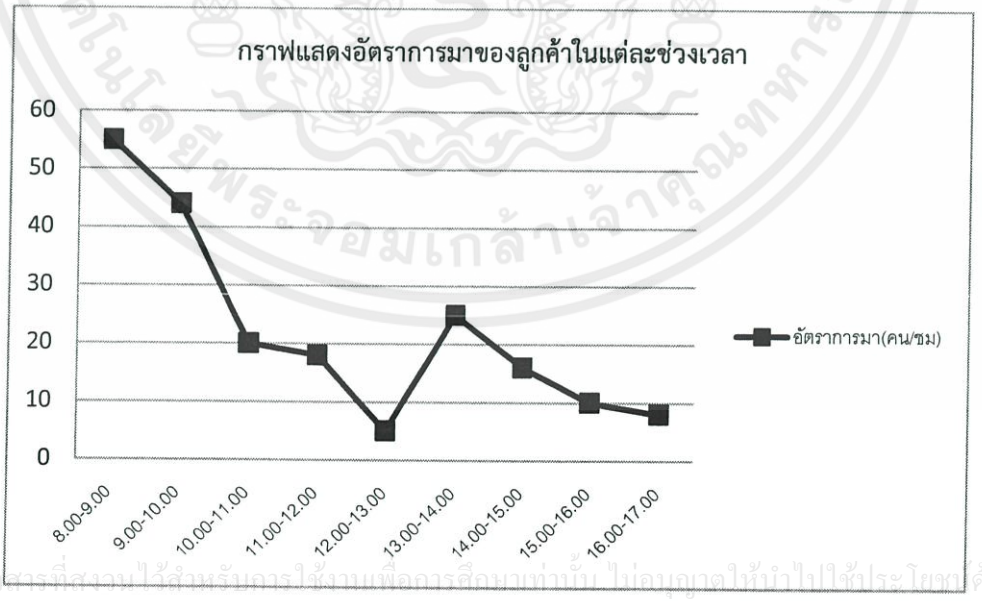



รูปที่ 4.6 กราฟแสดงความสัมพันธ์ระหว่างเวลารอคอยกับเวลาการมาถึงของลูกค้าของแถวคอยพลวัตแบบหลายช่องทาง-ขั้นตอนเดียว (ธนาคาร)

จากรูปที่ 4.6 แสดงความสัมพันธ์ระหว่างเวลารอคอยกับเวลาการมาถึงของลูกค้าของแถวคอยพลวัตแบบช่องทางเดียว-หลายขั้นตอน (ห้องจ่ายยา) ค่าเวลารอคอยเฉลี่ยของการบริการทั้งวันจากโปรแกรมจำลองสถานการณ์และโปรแกรมคำนวณเวลารอคอยตาราง ผ 3 เท่ากับ 1.481และ 1.638 นาที ตามลำดับ โดยที่มีความแตกต่างของค่าเฉลี่ยเวลารอคอยทั้งสองเท่ากับ 15.41% จากความสัมพันธ์ดังกล่าวสามารถวิเคราะห์ได้ว่า กราฟที่ได้ในช่วงแรกเวลารอคอยจากโปรแกรมประมาณเวลารอคอยจะมีค่าสูงกว่าค่าที่ได้จากโปรแกรม Arena เล็กน้อยซึ่งเกิดจากการประมาณค่าของข้อมูลในอดีตที่โปรแกรมนำมาใช้ในการคำนวณแต่ยังความสัมพันธ์กันเนื่องจากแนวโน้มของเวลารอคอยที่ค่อยๆเพิ่มสูงขึ้นและลดลงตามเวลาการถึงของลูกค้าเป็นไปในทิศทางเดียวกัน อีกทั้งเมื่อจำนวนลูกค้าเพื่อมากขึ้นในช่วงเวลาต่างๆกราฟทั้งสองก็จะขึ้นและลดลงในช่วงเวลาเดียวกัน ซึ่งเมื่อเปรียบเทียบกับกราฟแสดงอัตราการมาถึงของลูกค้าดังรูปที่ 4.5 จะเป็นไปในทิศทางเดียวกันคือ ลูกค้าจะเพิ่มสูงขึ้นในช่วงเวลา 11.00 น.-12.00 น. ส่งผลให้เกิดเวลารอคอยทีเพิ่มขึ้นและค่อยๆลดลงตามลำดับ

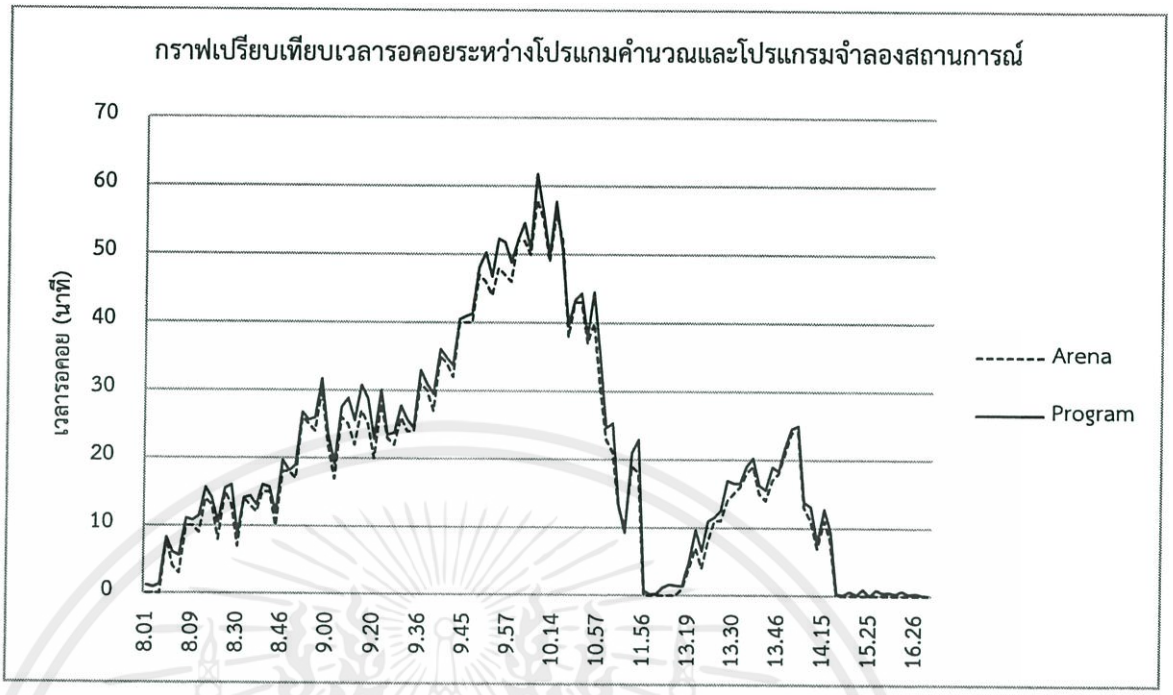
## 4.4 กราฟและการวิเคราะห์ผลลัพธ์ที่ได้จากแถวคอยพลวัตแบบหลายช่องทาง–หลายขั้นตอน (ห้องเจาะเลือด)

ดังที่ได้กล่าวในหัวข้อ 4.1 ได้ ซึ่งรูปแบบการให้บริการแบบหลายช่องทาง-หลายขั้นตอน (ห้องเจาะเลือด) จะกำหนดเวลาในการให้บริการไว้ที่ 9 นาทีต่อคนตามที่กล่าวไว้ในหัวข้อ 2.3.3 เวลาการมาถึงของลูกค้าและเวลารอคอยโดยประมาณที่ได้แสดงดังตารางที่ ผ 4 นำเวลารอคอยที่ได้จากตารางดังกล่าวมาสร้างกราฟแสดงความสัมพันธ์ระหว่างเวลารอคอยกับเวลาการมาถึงของลูกค้า เพื่อเปรียบเทียบแนวโน้มของผลลัพธ์ที่ได้ว่ามีความสัมพันธ์ในทิศทางเดียวกันหรือไม่ แสดงดังรูปที่ 4.8






รูปที่ 4.7 กราฟแสดงอัตราการมาของลูกค้าในแต่ละช่วงเวลา (Multi Channels - Multi Servers)




รูปที่ 4.8 กราฟแสดงความสัมพันธ์ระหว่างเวลารอคอยกับเวลาการมาถึงของลูกค้าของแถวคอยพลวัตแบบหลายช่องทาง-หลายขั้นตอน (ห้องเจาะเลือด)

จากรูปที่ 4.8 แสดงความสัมพันธ์ระหว่างเวลารอคอยกับเวลาการมาถึงของลูกค้าของแถวคอยพลวัตแบบช่องทางเดียว-หลายขั้นตอน (ห้องจ่ายยา) ค่าเวลารอคอยเฉลี่ยของการบริการทั้งวันจากโปรแกรมจำลองสถานการณ์และโปรแกรมคำนวณเวลารอคอยตาราง ผ 4 เท่ากับ 19.183 และ 20.657 นาที ตามลำดับ โดยที่มีความแตกต่างของค่าเฉลี่ยเวลารอคอยทั้งสองเท่ากับ 7.702% จากความสัมพันธ์ดังกล่าวสามารถวิเคราะห์ได้ว่ามีความสัมพันธ์กันเนื่องจากแนวโน้ม และการช่วงการขึ้น-ลงของเวลารอคอยตามเวลาการมาถึงของลูกค้าเป็นไปในทิศทางเดียวกัน ค่าเวลารอคอยของลูกค้ามีความใกล้เคียงหรือเท่ากันในบางช่วงเวลาและไม่พบค่าผิดปกติ (พุ่งสูงหรือลดต่ำเกินเมื่อเทียบกับค่าที่ได้จากอีกโปรแกรม) ที่จุดใดๆ เมื่อเปรียบเทียบกับกราฟแสดงอัตราการมาของลูกค้าในแต่ละช่วงดังรูปที่ 4.7 เวลาพบว่ามีความสัมพันธ์กันคือชั่วโมงแรกมีลูกค้าจำนวนมากจนค่อยๆเกิดการสะสมของเวลารอคอยอย่างช้าๆและค่อยๆลดลงตามลำดับ

ซึ่งจากผลการดำเนินการทั้งหมด สามารถสรุปความสามารถและศักยภาพความน่าเชื่อถือของโปรแกรมประมาณเวลารอคอยโดยตัวแบบแถวคอยพลวัตในบทต่อไป

# บทที่5
# สรุปและวิจารณ์

ในการพัฒนาโปรแกรมสำหรับประมาณเวลารอคอยสามารถสรุปและวิจารณ์ผลลัพธ์ที่ได้จากการทดลองและเปรียบเทียบกับผลลัพธ์ที่ได้จากโปรแกรมจำลองสถานการณ์ (Arena) โดยสามารถสรุปได้ดังนี้

## 5.1 สรุป

การพัฒนาโปรแกรมประมาณเวลารอคอยมีแนวคิดมาจากการเกิดแถวคอยในการบริการต่างๆ โดยที่ผู้เข้ารับบริการไม่สามารถทราบเวลารอคอยในการเข้ารับบริการ ซึ่งในแต่ละช่วงเวลามีอัตราการมาของลูกค้าที่แตกต่างกัน จากเหตุผลข้างต้นการประมาณเวลารอคอยของลูกค้าในช่วงเวลาต่างๆได้ ย่อมส่งผลดีต่อตัวผู้ให้และรับบริการในด้านการบริหารจัดการเวลาให้เกิดประโยชน์ต่อตนเองมากที่สุด โดยข้อมูลที่ใช้ในการคำนวณคือข้อมูลของเวลาการมาถึงของลูกค้า เวลาในการให้บริการ และเวลาออกจากระบบของลูกค้า โดยข้อมูลถูกสร้างมาจากอัตราการมาของลูกค้าและอัตราในการให้บริการซึ่งมีรูปแบบการแจกแจงแบบปัวร์ซองที่สร้างข้อมูลเก็บไว้อยู่ในแต่ละรูปแบบการบริการ และช่วงเวลาของการบริการเพื่อใช้ในการคำนวณหาอัตราการมาของลูกค้าและอัตราการให้บริการ เพื่อใช้ในการคำนวณในสมการหาเวลารอคอยของแต่ละรูปแบบการบริการ

ในการพัฒนาโปรแกรมคำนวณเวลารอคอยกำหนดให้โปรแกรมสามารถเลือกการคำนวณด้วยรูปแบบการบริการ 4 แบบ คือ

1. Single Channel Single Server สามารถเลือก ได้ 1 ช่องบริการ
2. Multi Channels Single Server สามารถเลือก ได้ 1 ช่องบริการ
3. Single Channels Multi Servers สามารถเลือก ได้ 1-3 ช่องบริการ
4. Multi Channels Multi Servers สามารถเลือก ได้ 1-3 ช่องบริการ

โดยคำนวณหาเวลารอคอยจากการนำอัตราการมาของลูกค้าและอัตราการให้บริการที่คำนวณจากเวลาการมาถึงของลูกค้า เวลาในการให้บริการ และเวลาออกจากระบบของลูกค้า มาคำนวณในสมการหาเวลารอคอยของแต่ละรูปแบบบริการโดยผลลัพธ์ที่ได้คือเวลารอคอยโดยเฉลี่ยของลูกค้าคนนั้นๆ โดยที่ความถูกต้องของเวลารอคอยที่ได้จากการคำนวณจากโปรแกรมขึ้นอยู่กับจำนวนข้อมูลที่นำมาคำนวณ โดยถ้าใช้จำนวนข้อมูลมากเท่าไรเวลารอคอยที่ได้จากการคำนวณจะยิ่งมีความถูกต้องและแม่นยำมากขึ้น โดยข้อมูลที่ใช้จำเป็นที่จะต้องมีความทันสมัยเสมอ เพราะเมื่อเวลาผ่านไปจะมีการเปลี่ยนแปลงของอัตราการมาและอัตราการให้บริการในแต่ละช่วงเวลาของการบริการ ซึ่งจะทำให้เวลารอคอยที่ได้จากการคำนวณมีความถูกต้องแม่นยำลดน้อยลง

การเปรียบเทียบเวลารอคอยระหว่างโปรแกรมประมาณเวลารอคอยกับการจำลองสถานการณ์นั้นเราจะใช้ข้อมูลเวลาการมาถึงของลูกค้าที่เข้ามารับบริการในวันหนึ่งของแต่ละรูปแบบการบริการในการทดสอบเพื่อเปรียบเทียบค่าเวลารอคอยที่ได้ของลูกค้าแต่ละคน โดยนำข้อมูลที่ได้จากการสร้างข้อมูลมาใช้เป็นชุดข้อมูลพื้นฐานในการเปรียบเทียบความน่าเชื่อถือของโปรแกรมที่เราพัฒนาขึ้นโดยมีแบบจำลองแยกตามลักษณะรูปแบบการบริการทั้ง 4 แบบตามที่กำหนด โดยสามารถสรุปและวิเคราะห์ผลลัพธ์ที่ได้จากการทดลองและเปรียบเทียบกับผลลัพธ์ที่ได้จากโปรแกรมจำลองสถานการณ์ (Arena) โดยสามารถสรุปได้ดังนี้ การพัฒนาโปรแกรมประมาณเวลารอคอยโดยตัวแบบแถวคอยพลวัต สามารถแสดงผลลัพธ์มีแนวโน้มใกล้เคียงกับโปรแกรมจำลองสถานการณ์ (Arena) โปรแกรมที่พัฒนาขึ้นจึงสามารถนำไปใช้งานได้จริงในงานบริการต่างๆที่มีรูปแบบการบริการตามที่โปรแกรมได้ถูกออกแบบไว้ โดยใช้ข้อมูลเวลาการมาถึงของลูกค้าที่สะสมไว้มาคำนวณหาอัตราการมาของลูกค้าโดยเฉลี่ยเนื่องจากแต่ละช่วงเวลาในการให้บริการนั้น ระยะห่างการมาถึงของลูกค้าจะไม่คงที่ (พลวัต) การข้อมูลการมาถึงของลูกค้าที่มากพอและสามารถรับข้อมูลใหม่ได้ ย่อมทำให้การประมาณเวลารอคอยใกล้เคียงความเป็นจริงและสามารถตอบสนองต่อความต้องการของทั้งผู้ให้และรับบริการได้ทันทีในโลกปัจจุบันที่มีการแข่งขันด้านงานบริการอย่างสูง

## 5.2 วิจารณ์

จากการพัฒนาโปรแกรมคำนวณหาเวลารอคอยมี ข้อดี ข้อเสียและแนวทางในการพัฒนาโปรแกรมคำนวณเวลารอคอยในอนาคตดังนี้

### 5.2.1 ข้อดีของโปรแกรมประมาณเวลารอคอย

1. ผู้ใช้งานสามารใช้งานโปรแกรมประมาณเวลารอคอยได้ง่ายและสะดวก
2. เวลารอคอยที่ได้จากการคำนวณเมื่อเปรียบเทียบกับโปรแกรมจำลองสถานการณ์มีแนวโน้มใกล้เคียงกัน
3. โปรแกรมประมาณเวลารอคอยสามารถเป็นตัวช่วยในการตัดสินใจของผู้เข้ารับบริการในการเข้ารับบริการหรือไม่รับ และผู้ใช้งานสามารถมองเห็นถึงปัญหาที่เกิดจากเวลารอคอยเพื่อการปรับปรุงการบริการให้ดีขึ้น
4. สร้างความพึงพอใจให้กับลูกค้า และส่งผลดีต่อการตอบสนองการเข้ารับบริการของลูกค้าในระยะยาวได้

### 5.2.2 ข้อเสียของโปรแกรมประมาณเวลารอคอย

1. โปรแกรมมีข้อจำกัดในส่วนของรูปแบบการให้บริการ ซึ่งโปรแกรมที่พัฒนามีเพียง 4 รูปแบบเท่านั้น
2. การคำนวณเวลารอคอยจำเป็นที่จะต้องใช้ข้อมูลในทางสถิติ ซึ่งผู้ใช้งานจำเป็นต้องมีการเก็บบันทึกไว้ก่อนใช้งานโปรแกรมประมาณเวลารอคอย
3. โปรแกรมอาจจะเกิดปัญหา เนื่องจากโปรแกรมไม่ได้รองรับในกรณีลูกค้าคนก่อนหน้าออกจากระบบโดยไม่เข้ารับบริการ ทำให้ลูกค้าคนต่อไปจะได้รับเวลารอคอยของคนก่อนหน้าแทนที่จะเป็นของตน

4. ในระยะสั้นจะไม่เห็นผลดีของการตอบสนองการเข้ารับบริการของลูกค้า เนื่องจากในกรณีที่เวลารอคอยของลูกค้าคลาดเคลื่อนจากปัจจุบันไปมากทำให้สูญเสียลูกค้า แต่ในระยะยาวแล้วลูกค้าจะเห็นความสำคัญของการทราบเวลารอคอยและจะกลับมาใช้บริการดังเดิม

### 5.2.3 แนวทางในการพัฒนาในอนาคต

แนวทางในการพัฒนาโปรแกรมประมาณเวลารอคอยในอนาคต มีดังนี้

1. เพิ่มความสามารถในการรองรับการคำนวณให้มีรูปแบบการบริการให้หลากหลายมากขึ้น
2. พัฒนาโปรแกรมให้สามารถครอบคลุมถึงปัญหาที่อาจจะเกิดขึ้นในระหว่างการใช้งาน เช่น ในกรณีลูกค้าคนก่อนหน้าออกจากระบบโดยไม่เข้ารับบริการ ทำให้ลูกค้าคนต่อไปจะได้รับเวลารอคอยของคนก่อนหน้าแทนที่จะเป็นของตน
3. พัฒนารูปแบบการใช้งานของโปรแกรมให้ใช้งานง่ายไม่ซับซ้อน สะดวกต่อผู้ใช้งาน มีรูปแบบหน้าต่างของโปรแกรมที่สวยงาม และมีประสิทธิภาพมากขึ้น
4. พัฒนาให้ใช้ได้กับการให้บริการแบบพลวัต

# หนังสืออ้างอิง


- รศ.ธีรวัฒน์ ประกอบผล, 2554. คู่มือการพัฒนาแอพพลิเคชันด้วย VisualC# 2010. พิมพ์ครั้งที่1.กรุงเทพฯ : สำนักพิมพ์ซิมพลิฟาย.
- รุ่งรัตน์ ภิสัชเพ็ญ , 2553. คู่มือสร้างแบบจำลองด้วยโปรแกรม Arena (ปรับปรุง). กรุงเทพฯ : สำนักพิมพ์ซีเอ็ดยูเคชั่น.
- รศ.สุทธิมา ชำนาญเวช , 2552. การวิจัยดำเนินงาน(Operation Research ). กรุงเทพฯ : สำนักพิมพ์วิทยพัฒน์
- Hillier Lieberman,2010. Introduction to Operation Research. แปลโดย รศ.ดร.พงศ์ชนัน เหลืองไพบูลย์. กรุงเทพฯ:สำนักพิมพ์ท้อป.

# ภาคผนวก

ตารางที่ ผ 1 ตารางแสดงตัวอย่างเวลาการมาถึงของลูกค้าและเวลารอคอยโดยประมาณของแถวคอยพลวัตแบบช่องทางเดียว-ขั้นตอนเดียว (ร้านอาหารตามสั่ง)

| Arrival Time | Waiting time(Min)/Arena | Waiting time(Min)/C# |
|---|---|---|
| 10.06 | 0 | 0 |
| 10.20 | 0 | 2.32 |
| 10.42 | 0 | 0.62 |
| 10.51 | 0 | 0.49 |
| 11.07 | 0 | 0.83 |
| 11.21 | 0 | 2.41 |
| 11.22 | 4 | 4.76 |
| 11.25 | 6 | 6.91 |
| 11.53 | 0 | 1.14 |
| 12.05 | 0 | 1.61 |
| 12.09 | 1 | 3.19 |
| 12.13 | 2 | 7.57 |
| 12.18 | 2 | 6.17 |
| 12.19 | 6 | 5.98 |
| 12.22 | 8 | 9 |
| 12.24 | 11 | 12.94 |
| 12.25 | 15 | 15.97 |
| 12.27 | 18 | 20.07 |
| 12.33 | 17 | 21.95 |
| 12.34 | 21 | 20.76 |
| 12.34 | 26 | 24.76 |
| 12.41 | 24 | 28.25 |
| 12.45 | 25 | 25.93 |
| 12.52 | 23 | 26.37 |
| 12.52 | 28 | 24.37 |
| 12.52 | 33 | 29.37 |
| 12.54 | 36 | 34.07 |
| 12.55 | 40 | 36.86 |
| 13.03 | 37 | 40.53 |
| 13.16 | 29 | 38.02 |

| Arrival Time | Waiting time(Min)/Arena | Waiting time(Min)/C# |
|---|---|---|
| 13.24 | 26 | 30.3 |
| 13.27 | 28 | 27.52 |
| 13.41 | 19 | 29.1 |
| 13.47 | 18 | 19.9 |
| 13.54 | 16 | 18.55 |
| 14.08 | 7 | 8 |
| 14.09 | 11 | 8.5 |
| 14.28 | 0 | 0.64 |
| 14.47 | 0 | 0.61 |
| 15.10 | 0 | 1.38 |
| 15.21 | 0 | 2.03 |
| 15.23 | 3 | 1.66 |
| 15.28 | 3 | 1.93 |
| 15.38 | 0 | 2.14 |
| 15.55 | 0 | 0.81 |
| 16.02 | 0 | 0.74 |
| 16.1 | 0 | 1.86 |
| 16.18 | 0 | 2 |
| 16.19 | 4 | 3.96 |
| 16.23 | 5 | 5.91 |
| 16.25 | 8 | 7.94 |
| 16.3 | 8 | 10.17 |
| 16.35 | 8 | 13.43 |
| 16.36 | 12 | 16.66 |
| 16.38 | 15 | 20.26 |
| 16.38 | 20 | 23.86 |
| 16.4 | 23 | 27.03 |
| 17.2 | 0 | 3.99 |
| 17.23 | 2 | 3.51 |
| 17.27 | 3 | 4.97 |
| 17.33 | 2 | 4.21 |
| 17.33 | 7 | 8.42 |
| 17.35 | 10 | 13.01 |
| 17.37 | 13 | 17.86 |

| Arrival Time | Waiting time(Min)/Arena | Waiting time(Min)/C# |
|---|---|---|
| 17.45 | 10 | 9.49 |
| 17.59 | 1 | 0.55 |
| 18.02 | 3 | 1.37 |
| 18.05 | 5 | 4.22 |
| 18.08 | 7 | 7.84 |
| 18.09 | 11 | 9.84 |
| 18.11 | 14 | 14.08 |
| 18.13 | 17 | 17.63 |
| 18.14 | 21 | 20.7 |
| 18.16 | 24 | 24.67 |
| 18.2 | 25 | 30.43 |
| 18.24 | 26 | 31.38 |
| 18.24 | 31 | 32.38 |
| 18.25 | 35 | 38.8 |
| 18.29 | 36 | 46.57 |
| 18.31 | 39 | 46.06 |
| 18.33 | 42 | 47.5 |
| 18.37 | 43 | 45.91 |
| 18.42 | 43 | 46.14 |
| 18.43 | 47 | 45.97 |
| 18.49 | 46 | 49.91 |
| 18.5 | 50 | 48.55 |
| 18.55 | 50 | 51.34 |
| 18.57 | 53 | 50.81 |
| 19.06 | 49 | 49 |
| 19.25 | 35 | 37 |
| 19.26 | 39 | 40.33 |
| 19.44 | 26 | 27 |
| 20.17 | 0 | 0.89 |
| 20.24 | 0 | 1.01 |
| 20.33 | 0 | 1.14 |

ตารางที่ ผ 2 ตารางแสดงตัวอย่างเวลาการมาถึงของลูกค้าและเวลารอคอยโดยประมาณของแถวคอยพลวัตแบบช่องทางเดียว-หลายขั้นตอน (ห้องจ่ายยา)

| Arrival Time | Waiting time(Min)/Arena | Waiting time(Min)/C# |
|---|---|---|
| 8.04 | 0 | 0.00 |
| 8.05 | 7 | 8.96 |
| 8.09 | 11 | 12.49 |
| 8.11 | 17 | 18.64 |
| 8.12 | 24 | 25.30 |
| 8.23 | 21 | 22.21 |
| 8.26 | 26 | 27.25 |
| 8.28 | 32 | 32.77 |
| 8.32 | 36 | 37.03 |
| 8.36 | 40 | 41.80 |
| 8.36 | 48 | 48.92 |
| 8.39 | 53 | 54.00 |
| 8.39 | 61 | 62.11 |
| 8.41 | 67 | 67.85 |
| 8.43 | 73 | 73.69 |
| 8.51 | 73 | 74.73 |
| 8.57 | 75 | 76.45 |
| 9.05 | 75 | 75.64 |
| 9.06 | 82 | 83.69 |
| 9.08 | 88 | 88.88 |
| 9.09 | 95 | 95.72 |
| 9.11 | 101 | 102.33 |
| 9.12 | 108 | 108.81 |
| 9.14 | 114 | 115.13 |
| 9.15 | 121 | 122.57 |
| 9.17 | 127 | 127.51 |
| 9.18 | 134 | 134.60 |
| 9.21 | 139 | 140.19 |
| 9.23 | 145 | 145.56 |
| 9.24 | 152 | 153.79 |

| Arrival Time | Waiting time(Min)/Arena | Waiting time(Min)/C# |
|---|---|---|
| 9.25 | 159 | 160.65 |
| 9.26 | 166 | 167.05 |
| 9.26 | 174 | 174.51 |
| 9.35 | 173 | 174.53 |
| 9.35 | 181 | 181.77 |
| 9.36 | 188 | 189.99 |
| 9.40 | 192 | 193.59 |
| 9.43 | 197 | 198.80 |
| 9.45 | 203 | 204.99 |
| 9.48 | 208 | 209.88 |
| 9.54 | 210 | 210.55 |
| 9.58 | 214 | 215.50 |
| 9.59 | 221 | 222.80 |
| 9.59 | 229 | 230.99 |
| 10.00 | 236 | 237.19 |
| 10.01 | 243 | 244.62 |
| 10.03 | 249 | 250.24 |
| 10.04 | 256 | 257.10 |
| 10.12 | 256 | 257.16 |
| 10.13 | 263 | 264.09 |
| 10.19 | 265 | 266.99 |
| 10.22 | 270 | 271.14 |
| 10.24 | 276 | 277.15 |
| 10.25 | 283 | 284.05 |
| 10.31 | 285 | 285.54 |
| 10.37 | 287 | 288.86 |
| 10.38 | 294 | 295.49 |
| 10.44 | 296 | 296.99 |
| 10.46 | 302 | 302.80 |
| 10.50 | 306 | 306.54 |
| 10.50 | 314 | 314.99 |
| 10.52 | 320 | 321.25 |
| 10.53 | 327 | 327.71 |
| 10.53 | 335 | 336.01 |

| Arrival Time | Waiting time(Min)/Arena | Waiting time(Min)/C# |
|---|---|---|
| 10.53 | 343 | 343.95 |
| 10.59 | 345 | 346.00 |
| 10.59 | 353 | 354.41 |

ตารางที่ ผ 3 ตารางแสดงตัวอย่างเวลาการมาถึงของลูกค้าและเวลารอคอยโดยประมาณของแถวคอยพลวัตแบบหลายช่องทาง - ขั้นตอนเดียว (ธนาคาร)

| Arrival Time | Waiting time(Min)/Arena | Waiting time(Min)/C# |
|---|---|---|
| 8.05 | 0 | 0.00 |
| 8.08 | 0 | 0.26 |
| 8.15 | 0 | 0.41 |
| 8.26 | 0 | 0.20 |
| 8.36 | 0 | 0.18 |
| 8.49 | 0 | 0.44 |
| 9.01 | 0 | 0.06 |
| 9.18 | 0 | 0.14 |
| 9.23 | 0 | 0.34 |
| 9.34 | 0 | 0.39 |
| 9.37 | 0 | 0.23 |
| 9.44 | 0 | 0.49 |
| 9.49 | 0 | 0.34 |
| 9.50 | 0 | 0.27 |
| 9.52 | 1 | 1.41 |
| 10.10 | 0 | 0.28 |
| 10.12 | 0 | 0.19 |
| 10.14 | 0 | 0.43 |
| 10.19 | 0 | 0.09 |
| 10.19 | 2 | 2.07 |
| 10.27 | 0 | 0.31 |
| 10.28 | 0 | 0.26 |
| 10.28 | 2 | 2.08 |
| 10.39 | 0 | 0.02 |

| Arrival Time | Waiting time(Min)/Arena | Waiting time(Min)/C# |
|---|---|---|
| 10.42 | 0 | 0.09 |
| 10.45 | 0 | 0.28 |
| 10.45 | 3 | 4.48 |
| 10.49 | 2 | 2.66 |
| 10.52 | 2 | 2.95 |
| 10.54 | 3 | 3.29 |
| 10.55 | 5 | 6.90 |
| 10.57 | 6 | 6.37 |
| 11.01 | 5 | 6.44 |
| 11.02 | 7 | 8.97 |
| 11.02 | 10 | 11.47 |
| 11.05 | 10 | 10.48 |
| 11.06 | 12 | 13.18 |
| 11.12 | 9 | 10.06 |
| 11.17 | 7 | 7.74 |
| 11.25 | 2 | 2.04 |
| 11.26 | 4 | 5.26 |
| 11.31 | 2 | 2.95 |
| 11.32 | 4 | 5.41 |
| 11.39 | 0 | 1.43 |
| 11.41 | 1 | 2.35 |
| 11.47 | 0 | 1.41 |
| 11.52 | 0 | 1.93 |
| 11.52 | 0 | 1.51 |
| 11.56 | 0 | 1.21 |
| 11.57 | 4 | 5.78 |
| 12.02 | 0 | 0.98 |
| 12.05 | 0 | 0.40 |
| 12.06 | 4 | 4.84 |
| 12.07 | 4 | 5.55 |
| 12.07 | 7 | 7.20 |
| 12.11 | 8 | 9.45 |
| 12.20 | 0 | 0.79 |
| 12.20 | 3 | 3.06 |

| Arrival Time | Waiting time(Min)/Arena | Waiting time(Min)/C# |
|---|---|---|
| 12.27 | 1 | 2.09 |
| 12.29 | 0 | 0.13 |
| 12.42 | 0 | 0.09 |
| 12.46 | 0 | 0.17 |
| 12.48 | 0 | 0.32 |
| 12.48 | 3 | 3.70 |
| 12.56 | 0 | 1.40 |
| 13.03 | 0 | 0.25 |
| 13.04 | 0 | 0.29 |
| 13.12 | 0 | 0.46 |
| 13.14 | 0 | 0.29 |
| 13.20 | 0 | 0.17 |
| 13.25 | 0 | 0.16 |
| 13.33 | 0 | 0.24 |
| 13.35 | 0 | 0.03 |
| 13.37 | 0 | 0.48 |
| 13.42 | 0 | 0.19 |
| 13.44 | 0 | 0.21 |
| 13.59 | 0 | 0.16 |
| 14.02 | 0 | 0.28 |
| 14.09 | 0 | 0.13 |
| 14.26 | 0 | 0.25 |
| 14.27 | 0 | 0.45 |
| 14.30 | 0 | 0.47 |
| 14.30 | 5 | 6.73 |
| 14.38 | 0 | 0.43 |
| 14.55 | 0 | 0.15 |
| 14.56 | 0 | 0.27 |
| 14.58 | 0 | 0.02 |
| 15.09 | 0 | 0.23 |
| 15.16 | 0 | 0.07 |
| 15.20 | 0 | 0.31 |
| 15.27 | 0 | 0.22 |
| 15.28 | 0 | 0.04 |

| Arrival Time | Waiting time(Min)/Arena | Waiting time(Min)/C# |
|---|---|---|
| 15.50 | 0 | 0.34 |
| 15.57 | 0 | |

ตารางที่ ผ 4 ตารางแสดงตัวอย่างเวลาการมาถึงของลูกค้าและเวลารอคอยโดยประมาณของแถวคอยพลวัตแบบหลายช่องทาง-หลายขั้นตอน (ห้องเจาะเลือด)

| Arrival Time | Waiting time(Min)/Arena | Waiting time(Min)/C# |
|---|---|---|
| 8.01 | 0 | 1.19 |
| 8.01 | 0 | 0.90 |
| 8.01 | 0 | 1.31 |
| 8.02 | 8 | 8.21 |
| 8.06 | 4 | 6.03 |
| 8.07 | 3 | 5.57 |
| 8.09 | 10 | 11.09 |
| 8.09 | 10 | 10.78 |
| 8.10 | 9 | 11.45 |
| 8.14 | 14 | 15.62 |
| 8.15 | 13 | 14.07 |
| 8.20 | 8 | 10.48 |
| 8.22 | 15 | 15.49 |
| 8.24 | 13 | 16.01 |
| 8.30 | 7 | 8.75 |
| 8.32 | 14 | 14.08 |
| 8.33 | 13 | 14.37 |
| 8.34 | 12 | 13.09 |
| 8.40 | 15 | 16.01 |
| 8.40 | 15 | 15.71 |
| 8.45 | 10 | 11.86 |
| 8.46 | 18 | 19.64 |
| 8.46 | 18 | 18.17 |
| 8.47 | 17 | 18.97 |
| 8.47 | 26 | 26.73 |

| Arrival Time | Waiting time(Min)/Arena | Waiting time(Min)/C# |
|---|---|---|
| 8.48 | 25 | 25.64 |
| 8.49 | 24 | 25.97 |
| 8.52 | 30 | 31.66 |
| 9.00 | 22 | 23.54 |
| 9.05 | 17 | 19.19 |
| 9.05 | 26 | 27.53 |
| 9.06 | 25 | 28.79 |
| 9.09 | 22 | 25.60 |
| 9.13 | 27 | 30.73 |
| 9.15 | 25 | 28.77 |
| 9.20 | 20 | 22.85 |
| 9.21 | 28 | 30.03 |
| 9.26 | 23 | 23.49 |
| 9.27 | 22 | 23.74 |
| 9.32 | 26 | 27.70 |
| 9.34 | 24 | 25.67 |
| 9.34 | 24 | 24.52 |
| 9.36 | 31 | 32.97 |
| 9.37 | 30 | 31.00 |
| 9.40 | 27 | 29.50 |
| 9.41 | 35 | 36.08 |
| 9.42 | 34 | 34.79 |
| 9.44 | 32 | 33.71 |
| 9.45 | 40 | 40.49 |
| 9.45 | 40 | 40.97 |
| 9.45 | 40 | 41.29 |
| 9.47 | 47 | 48.13 |
| 9.48 | 46 | 50.29 |
| 9.50 | 44 | 46.77 |
| 9.55 | 48 | 52.32 |
| 9.56 | 47 | 51.78 |
| 9.57 | 46 | 48.88 |
| 10.00 | 52 | 52.01 |
| 10.00 | 52 | 54.58 |

| Arrival Time | Waiting time(Min)/Arena | Waiting time(Min)/C# |
|---|---|---|
| 10.02 | 50 | 50.69 |
| 10.03 | 58 | 61.79 |
| 10.06 | 55 | 56.20 |
| 10.12 | 49 | 49.68 |
| 10.14 | 56 | 57.75 |
| 10.18 | 52 | 50.62 |
| 10.32 | 38 | 39.59 |
| 10.36 | 43 | 43.34 |
| 10.36 | 43 | 44.33 |
| 10.42 | 37 | 37.92 |
| 10.48 | 40 | 44.51 |
| 10.57 | 31 | 35.62 |
| 11.05 | 23 | 24.66 |
| 11.16 | 21 | 25.33 |
| 11.24 | 13 | 13.27 |
| 11.27 | 10 | 9.27 |
| 11.27 | 19 | 20.98 |
| 11.28 | 18 | 22.90 |
| 11.56 | 0 | 0.71 |
| 12.08 | 0 | 0.18 |
| 12.47 | 0 | 0.25 |
| 13.10 | 0 | 1.19 |
| 13.14 | 0 | 1.61 |
| 13.18 | 0 | 1.45 |
| 13.18 | 1 | 1.39 |
| 13.19 | 4 | 4.88 |
| 13.20 | 7 | 9.64 |
| 13.24 | 4 | 6.57 |
| 13.24 | 8 | 10.95 |
| 13.25 | 11 | 11.52 |
| 13.26 | 11 | 12.55 |
| 13.27 | 14 | 16.94 |
| 13.30 | 15 | 16.51 |
| 13.30 | 16 | 16.49 |

| Arrival Time | Waiting time(Min)/Arena | Waiting time(Min)/C# |
|---|---|---|
| 13.32 | 18 | 19.01 |
| 13.35 | 19 | 20.25 |
| 13.40 | 15 | 16.24 |
| 13.45 | 14 | 15.55 |
| 13.46 | 17 | 18.91 |
| 13.46 | 18 | 18.30 |
| 13.47 | 21 | 21.87 |
| 13.48 | 24 | 24.55 |
| 13.49 | 24 | 24.97 |
| 14.04 | 13 | 13.82 |
| 14.10 | 11 | 13.12 |
| 14.15 | 7 | 7.69 |
| 14.15 | 11 | 12.74 |
| 14.22 | 8 | 9.51 |
| 14.32 | 0 | 0.29 |
| 14.37 | 0 | 0.15 |
| 14.46 | 0 | 0.62 |
| 15.01 | 0 | 0.16 |
| 15.18 | 0 | 0.94 |
| 15.25 | 0 | 0.01 |
| 15.34 | 0 | 0.83 |
| 15.39 | 0 | 0.44 |
| 15.49 | 0 | 0.54 |
| 16.03 | 0 | 0.30 |
| 16.08 | 0 | 0.72 |
| 16.11 | 0 | 0.26 |
| 16.26 | 0 | 0.40 |
| 16.31 | 0 | 0.14 |
| 16.40 | 0 | 0.06 |

**รหัสโปรแกรม**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Collections;

namespace project_waiting_time
{
    public static class ClassCalculate
    {
        public static void singleServer_calculate()
        {
            ArrayList arrText1 = new ArrayList();
            ArrayList arrText2 = new ArrayList();
            ArrayList arrText3 = new ArrayList();
            double a, b;
            double intervalTime = 0;
            double total_IntervalTime = 0;
            double avgIT = 0, lamda = 0;
            double U = 0;
            double LastTime = 0;
            double WqSS = 0;
            for (int j = 29; j >= 0; j--)
            {
                arrText1.Clear();
                arrText2.Clear();
                arrText3.Clear();
                a = 0; b = 0;
                int i = -1;
```

```
int n = 0;
if (!File.Exists(ClassGlobal.pathModel + @"\เวลามา\" + ClassGlobal.fileName[j] + ".txt"))
{
    MessageBox.Show("File " + ClassGlobal.fileName[j] + ".txt" + " not found!");
    return;
}
else
{
    StreamReader objReader1 = new StreamReader(ClassGlobal.pathModel + @"\เวลามา\" +
ClassGlobal.fileName[j] + ".txt");
    string tempRead1 = "";
    while ((tempRead1 = objReader1.ReadLine()) != null)
    {
        i++;
        arrText1.Add(tempRead1);
        if (double.Parse(arrText1[i].ToString()) > ClassGlobal.enterTime && n == 0)
        {
            if (i > 0) a = Convert.ToDouble(arrText1[i - 1]);
            if (a != 0) b = Convert.ToDouble(arrText1[i]);
            n = 1;
        }
    }
    objReader1.Close();
}
if (!File.Exists(ClassGlobal.pathModel + @"\เวลาเข้าบริการ\" + ClassGlobal.fileName[j] + ".txt"))
{
    MessageBox.Show("File " + ClassGlobal.fileName[j] + ".txt" + " not found!");
    return;
}
else
{
    StreamReader objReader2 = new StreamReader(ClassGlobal.pathModel + @"\เวลาเข้า
บริการ\" + ClassGlobal.fileName[j] + ".txt");
    string tempRead2 = "";
    while ((tempRead2 = objReader2.ReadLine()) != null)
    {
```

```
                arrText2.Add(tempRead2);
            }
            objReader2.Close();
        }
        if (!File.Exists(ClassGlobal.pathModel + @"\เวลาออก\" + ClassGlobal.fileName[j] + ".txt"))
        {
            MessageBox.Show("File " + ClassGlobal.fileName[j] + ".txt" + " not found!");
            return;
        }
        else
        {
            StreamReader objReader3 = new StreamReader(ClassGlobal.pathModel + @"\เวลาออก\"
+ ClassGlobal.fileName[j] + ".txt");
            string tempRead3 = "";
            while ((tempRead3 = objReader3.ReadLine()) != null)
            {
                arrText3.Add(tempRead3);
            objReader3.Close();
            }
        intervalTime = b - a;
        total_IntervalTime += intervalTime;

        if (j == 0)
        {
            avgIT = total_IntervalTime / 30;
            lamda = 0.60 / avgIT;
        }
        for (int m = 0; m < arrText2.Count; m++)
        {
            int chkLenght3 = 0, chkLenght2 = 0;
            double temparrText3 = 0, temparrText2 = 0;
            chkLenght3 = arrText3[m].ToString().Length; chkLenght2 = arrText2[m].ToString().Length;
            if (chkLenght3 == 4) temparrText3 =
Convert.ToDouble(arrText3[m].ToString().Substring(2, 2));
            if (chkLenght3 == 5) temparrText3 =
Convert.ToDouble(arrText3[m].ToString().Substring(3, 2));
```

```
                if (chkLenght2 == 4) temparrText2 =
Convert.ToDouble(arrText2[m].ToString().Substring(2, 2));
                if (chkLenght2 == 5) temparrText2 =
Convert.ToDouble(arrText2[m].ToString().Substring(3, 2));
                if (temparrText3 < temparrText2)
                {
                    if (temparrText3 == 0)
                        U += ((Convert.ToDouble(arrText3[m].ToString()) + 0.60) - 1.00 -
Convert.ToDouble(arrText2[m].ToString()));
                    else
                        U += ((Convert.ToDouble(arrText3[m].ToString())) -
Convert.ToDouble(arrText2[m].ToString())) - 0.40;
                }
                else
                    U += (Convert.ToDouble(arrText3[m].ToString()) -
Convert.ToDouble(arrText2[m].ToString()));
            }
            U = U / (arrText2.Count);
            if (j == 0)
            {
                U = 0.6 / U;
            }
        }
        if (lamda < U)
        {
            WqSS = lamda / (U * (U - lamda));
            WqSS = WqSS * 60;
        }
        else
        {
            DateTime dtToday = new DateTime();
            dtToday = DateTime.Today.Date;
            string Today = String.Format("{0:d_M_yyyy}", dtToday);
            if (!File.Exists(ClassGlobal.pathModel + @"\เวลาเข้าบริการ\" + Today + ".txt"))
            {
                return;
```

```
            }
            else
            {
                StreamReader objReaderLastTime = new StreamReader(ClassGlobal.pathModel + @"\
เวลาเข้าบริการ\" + Today + ".txt");
                string tempLastTime = "";
                while ((tempLastTime = objReaderLastTime.ReadLine()) != null)
                {
                    LastTime = double.Parse(tempLastTime);
                }
                objReaderLastTime.Close();
            }
            WqSS = (LastTime + avgIT) - ClassGlobal.enterTime;
        }
        ClassGlobal.Wq = WqSS;
        ClassGlobal.checkLowerQueueDuration.Add(ClassGlobal.Wq);
        ClassGlobal.checkLowerQueueTime.Add(ClassGlobal.enterTime);
        Check_QtimeNow();
    }
    public static void multiServer_calculate()
    {
        ArrayList arrText1 = new ArrayList();
        ArrayList arrText2 = new ArrayList();
        ArrayList arrText3 = new ArrayList();
        double a = 0, b = 0;
        double intervalTime = 0;
        double total_IntervalTime = 0;
        double avgIT = 0, lamda = 0;
        double U = 0;
        double LastTime = 0;
        double WqMS = 0;
        for (int j = 29; j >= 0; j--)
        {
            arrText1.Clear();
            arrText2.Clear();
            arrText3.Clear();
```

```
            int i = -1;
            int n=0;
            if (!File.Exists(ClassGlobal.pathModel + @"\เวลามา\" + ClassGlobal.fileName[j] + ".txt"))
            {
                MessageBox.Show("File " + ClassGlobal.fileName[j] + ".txt" + " not found!");
                return;
            }
            else
            {
                StreamReader objReader1 = new StreamReader(ClassGlobal.pathModel + @"\เวลามา\" +
ClassGlobal.fileName[j] + ".txt");
                string tempRead1 = "";
                while ((tempRead1 = objReader1.ReadLine()) != null)
                {
                    i++;
                    arrText1.Add(tempRead1);
                    if (double.Parse(arrText1[i].ToString()) > ClassGlobal.enterTime && n == 0)
                    {
                        if (i > 0) a = Convert.ToDouble(arrText1[i - 1]);
                        if (a != 0) b = Convert.ToDouble(arrText1[i]);
                        n = 1;
                    }
                }
                objReader1.Close();
            }
            if (!File.Exists(ClassGlobal.pathModel + @"\เวลาเข้าบริการ\" + ClassGlobal.fileName[j] + ".txt"))
            {
                MessageBox.Show("File " + ClassGlobal.fileName[j] + ".txt" + " not found!");
                return;
            }
            else
            {
                StreamReader objReader2 = new StreamReader(ClassGlobal.pathModel + @"\เวลาเข้า
บริการ\" + ClassGlobal.fileName[j] + ".txt");
                string tempRead2 = "";
                while ((tempRead2 = objReader2.ReadLine()) != null)
```

```
            {
                arrText2.Add(tempRead2);
            }
            objReader2.Close();
        }
        if (!File.Exists(ClassGlobal.pathModel + @"\เวลาออก\" + ClassGlobal.fileName[j] + ".txt"))
        {
            MessageBox.Show("File " + ClassGlobal.fileName[j] + ".txt" + " not found!");
            return;
        }
        else
        {
            StreamReader objReader3 = new StreamReader(ClassGlobal.pathModel + @"\เวลาออก\"
+ ClassGlobal.fileName[j] + ".txt");
            string tempRead3 = "";
            while ((tempRead3 = objReader3.ReadLine()) != null)
            {
                arrText3.Add(tempRead3);
            }
            objReader3.Close();
        }
        intervalTime = b - a;
        total_IntervalTime += intervalTime;

        if (j == 0)
        {
            avgIT = total_IntervalTime / 30;
            lamda = 0.60 / avgIT;
        }
        for (int m = 0; m < arrText2.Count; m++)
        {
            int chkLenght3 = 0, chkLenght2 = 0;
            double temparrText3 = 0, temparrText2 = 0;
            chkLenght3 = arrText3[m].ToString().Length; chkLenght2 = arrText2[m].ToString().Length;
            if (chkLenght3 == 4) temparrText3 =
Convert.ToDouble(arrText3[m].ToString().Substring(2, 2));
```

```
                if (chkLenght3 == 5) temparrText3 =
Convert.ToDouble(arrText3[m].ToString().Substring(3, 2));
                if (chkLenght2 == 4) temparrText2 =
Convert.ToDouble(arrText2[m].ToString().Substring(2, 2));
                if (chkLenght2 == 5) temparrText2 =
Convert.ToDouble(arrText2[m].ToString().Substring(3, 2));
                if (temparrText3 < temparrText2)
                {
                    if (temparrText3 == 0)
                        U += ((Convert.ToDouble(arrText3[m].ToString())+ 0.60) - 1.00 -
Convert.ToDouble(arrText2[m].ToString()));
                    else
                        U += ((Convert.ToDouble(arrText3[m].ToString())) -
Convert.ToDouble(arrText2[m].ToString())) - 0.40;
                }
                else
                    U += (Convert.ToDouble(arrText3[m].ToString()) -
Convert.ToDouble(arrText2[m].ToString()));
            }
            U = U / (arrText2.Count);
            if (j == 0)
            {
                U = 0.6 / U;
            }

        }
        double p, s, L, sU, P0, LU, LUn = 1, LUs = 1, LUsFact = 0, sumN = 0, x, y, z, zz;
        L = lamda;
        LU = L / U;
        s = ClassGlobal.ServiceCH;
        sU = s * U;
        if (L < sU)
        {
            p = L / sU;
            int sFact = 1, nFact = 1;
            for (int n = 1; n <= s - 1; n++)
```

```
        {
            nFact = nFact * n;

            LUn *= LU;
            sumN = LUn / nFact + 1;
        }
        for (int i = 1; i <= s; i++)
        {
            LUs *= LU;
            sFact = sFact * i;
            LUsFact = LUs / sFact;
        }
        x = sU - L;
        y = sU / x;
        z = LUsFact * y;
        zz = sumN + z;
        P0 = 1 / zz;
        double Lq, p1, p2, x1, y1, z1;
        p1 = 1 - p;
        p2 = p1 * p1;
        x1 = sFact * p2;
        y1 = LUs * p;
        z1 = y1 / x1;
        Lq = P0 * z1;
        WqMS = (Lq / L) * 60;
    }
    else
    {
        DateTime dtToday = new DateTime();
        dtToday = DateTime.Today.Date;
        string Today = String.Format("{0:d_M_yyyy}", dtToday);
        if (!File.Exists(ClassGlobal.pathModel + @"\เวลาเข้าบริการ\" + Today + ".txt"))
        {
            return;
        }
        else
```

```
            {
                StreamReader objReaderLastTime = new StreamReader(ClassGlobal.pathModel + @"\
เวลาเข้าบริการ\" + Today + ".txt");
                string tempLastTime = "";
                while ((tempLastTime = objReaderLastTime.ReadLine()) != null)
                {
                    LastTime = double.Parse(tempLastTime);
                }
                objReaderLastTime.Close();
            }
            WqMS = (LastTime + avgIT) - ClassGlobal.enterTime;
        }
        ClassGlobal.Wq = WqMS;
        ClassGlobal.checkLowerQueueDuration.Add(ClassGlobal.Wq);
        ClassGlobal.checkLowerQueueTime.Add(ClassGlobal.enterTime);
        Check_QtimeNow();
    }

    public static void Check_QtimeNow()
    {
        string TimeNow = DateTime.Now.ToShortTimeString().Replace(':', '.');
        if ((ClassGlobal.checkLowerQueueDuration.Count == 0) ||
(ClassGlobal.checkLowerQueueDuration.Count == 1))
        {
            return;
        }
        else
        {
            int index1 = ClassGlobal.checkLowerQueueDuration.Count;
            string timeQ1 = Convert.ToString(ClassGlobal.checkLowerQueueTime[index1 - 2]);
            string timeQ2 = Convert.ToString(ClassGlobal.checkLowerQueueTime[index1 - 1]);

            string MyString = Convert.ToDouble(timeQ1).ToString("F2");
            MyString = MyString.Replace('.', ':');
            string Mix = DateTime.Now.ToShortDateString() + " " + MyString + ":00";
            DateTime dt1 = Convert.ToDateTime(Mix);
```

```
                double QMin1 = Convert.ToDouble(ClassGlobal.checkLowerQueueDuration[index1 - 2]);


                string MyString2 = Convert.ToDouble(timeQ2).ToString("F2");
                MyString2 = MyString2.Replace('.', ':');
                string Mix2 = DateTime.Now.ToShortDateString() + " " + MyString2 + ":00";
                DateTime dt2 = Convert.ToDateTime(Mix2);
                double QMin2 = Convert.ToDouble(ClassGlobal.checkLowerQueueDuration[index1 - 1]);

                if ((dt1.AddMinutes(QMin1)) < (dt2.AddMinutes(QMin2)))
                {
                    return;
                }
                else
                {
                    dt1 = dt1.AddMinutes(QMin1);
                    TimeSpan TS = dt1 - dt2;
                    string stringTS = TS.ToString().Replace(':', '.');
                    string resultMin = stringTS.Substring(3, 5);
                    ClassGlobal.Wq = double.Parse(resultMin) + double.Parse(QMin2.ToString());
                    ClassGlobal.checkLowerQueueDuration.RemoveAt(index1-1);
                    ClassGlobal.checkLowerQueueDuration.Add(ClassGlobal.Wq);
                }
            }
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Collections;

namespace project_waiting_time
{
    static class ClassGlobal
    {
        public static int roomService;
        public static ArrayList fileName = new ArrayList();
        public static double Wq ;
        public static double enterTime;
        public static string pathModel;
        public static double ServiceCH;
        public static ArrayList checkLowerQueueTime = new ArrayList();
        public static ArrayList checkLowerQueueDuration = new ArrayList();
        public static string showBoard;
        public static string [] arrQValues = new string[3];
        public static int nextQ;
        static string fileN = "status.txt";
        static string pathStatusBoard = @"D:\ProjectFinal\Data\StatusBoard\" + fileN;
        public static frmBoard frmBoard = new frmBoard();
        public static string MessageRemoveQ;

        static public void Load_StatusData()
        {
            if(!File.Exists(pathStatusBoard))
            {
                StreamWriter SW;
```

```
            SW = File.CreateText(pathStatusBoard);
            SW.Close();
        }
        string tempReader="";
        StreamReader objReader = new StreamReader(pathStatusBoard);
        int i = 0;
        while ((tempReader = objReader.ReadLine()) != null)
            {
                arrQValues[i]=tempReader;
                i++;
            }
            objReader.Close();
    }

    static public void Save_StatusData()
    {
        if (!File.Exists(pathStatusBoard))
        {
            StreamWriter SW;
            SW = File.CreateText(pathStatusBoard);
            SW.Close();
        }
        StreamWriter objWriter = new StreamWriter(pathStatusBoard);
        for (int i = 0; i < 3; i++)
        {
            objWriter.WriteLine(arrQValues[i]);
        }
        objWriter.Close();
    }

    static public void Show_NextQueue()
    {
        string DateNow = DateTime.Now.ToShortDateString().Replace('/', '_');
        string pathQ_Out = @"D:\ProjectFinal\Data\Queue\2\" + DateNow + ".txt";
        ArrayList arrRead2 = new ArrayList();
```

```
if (!File.Exists(pathQ_Out))
{
    StreamWriter SW;
    SW = File.CreateText(pathQ_Out);
    SW.Close();
}
StreamReader objReader2 = new StreamReader(pathQ_Out);
string tempRead2 = "";
while ((tempRead2 = objReader2.ReadLine()) != null)
{
    arrRead2.Add(int.Parse(tempRead2));
    nextQ = Convert.ToInt32(tempRead2);
}
objReader2.Close();
nextQ += 1;
string pathQ_In = @"D:\ProjectFinal\Data\Queue\1\" + DateNow + ".txt";
ArrayList arrRead1 = new ArrayList();
if (!File.Exists(pathQ_In))
{
    StreamWriter SW;
    SW = File.CreateText(pathQ_In);
    SW.Close();
}
StreamReader objReader1 = new StreamReader(pathQ_In);
string tempRead1 = "";
while ((tempRead1 = objReader1.ReadLine()) != null)
{
    arrRead1.Add(int.Parse(tempRead1));
}
objReader1.Close();
if (arrRead1.Count > arrRead2.Count)
{
    for (int i = 0; i < arrRead1.Count; i++)
    {
        if (int.Parse(arrRead1[i].ToString()) == nextQ)
        {
```

```
                nextQ = int.Parse(arrRead1[i].ToString());
                break;
            }
            if (int.Parse(arrRead1[i].ToString()) > nextQ)
            {
                nextQ = int.Parse(arrRead1[i].ToString());
                break;
            }
        }
    }
}

static public void Remove_Queue()
{
    string DateNow = DateTime.Now.ToShortDateString().Replace('/', '_');
    string pathQ_In = @"D:\ProjectFinal\Data\Queue\1\" + DateNow + ".txt";
    ArrayList arrRemove1 = new ArrayList();

    if (!File.Exists(pathQ_In))
    {
        return;
    }

    StreamReader objReaderRemove = new StreamReader(pathQ_In);
    string tempReadRemove = "";
    while ((tempReadRemove = objReaderRemove.ReadLine()) != null)
    {
        arrRemove1.Add(int.Parse(tempReadRemove));
    }
    objReaderRemove.Close();
    for (int index = 0; index < arrRemove1.Count; index++)
    {
        if (int.Parse(arrRemove1[index].ToString()) == nextQ)
        {
            arrRemove1.RemoveAt(index);
            StreamWriter objWriter = new StreamWriter(pathQ_In);
```

```
                for (int i = 0; i < arrRemove1.Count; i++)
                {
                    objWriter.WriteLine(arrRemove1[i]);
                }
                objWriter.Close();
                nextQ += 1;
                MessageRemoveQ = "Yes";
                break;
            }
            else
                MessageRemoveQ = "No";
        }

    }
  }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Collections;
namespace project_waiting_time
{

    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            ClassGlobal.Show_NextQueue();
            ClassGlobal.showBoard = "close";
            ClassGlobal.fileName.Clear();
            DateTime dtTempDate = new DateTime();
            for (int i = 1; i < 31; i++)
            {
                if (dtTempDate != null)
                {
                    dtTempDate = DateTime.Today.Date.AddDays(-i);
                    ClassGlobal.fileName.Add(String.Format("{0:d_M_yyyy}", dtTempDate));
                }
            }
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            timer1.Start();
        }
```

```
private void timer1_Tick(object sender, EventArgs e)
{
    lbTimer.Text = DateTime.Now.ToLongTimeString();
    lbDate.Text = DateTime.Now.ToLongDateString();
}

private void rbtn_SS_CheckedChanged(object sender, EventArgs e)
{
    ClassGlobal.pathModel = @"D:\ProjectFinal\Data\1SingleCSigleServer";
    comboBox1.Items.Clear();
    comboBox1.Items.Insert(0, "กรุณาเลือก Service CH");
    comboBox1.Items.Insert(1, "Channel 1");
    comboBox1.SelectedIndex = 0;
}

private void rbtnSM_CheckedChanged(object sender, EventArgs e)
{
    ClassGlobal.pathModel = @"D:\ProjectFinal\Data\2SingleCMultiServer";
    comboBox1.Items.Clear();
    comboBox1.Items.Insert(0, "กรุณาเลือก Service CH");
    comboBox1.Items.Insert(1, "Channel 1");
    comboBox1.Items.Insert(2, "Channel 2");
    comboBox1.Items.Insert(3, "Channel 3");
    comboBox1.SelectedIndex = 0;
}

private void rbtnMS_CheckedChanged(object sender, EventArgs e)
{
    ClassGlobal.pathModel = @"D:\ProjectFinal\Data\3MultiCSingleServer";
    comboBox1.Items.Clear();
    comboBox1.Items.Insert(0, "กรุณาเลือก Service CH");
    comboBox1.Items.Insert(1, "Channel 1");
    comboBox1.SelectedIndex = 0;
}
```

```
private void rbtnMM_CheckedChanged(object sender, EventArgs e)
{
    ClassGlobal.pathModel = @"D:\ProjectFinal\Data\4MultiCMultiServer";
    comboBox1.Items.Clear();
    comboBox1.Items.Insert(0, "กรุณาเลือก Service CH");
    comboBox1.Items.Insert(1, "Channel 1");
    comboBox1.Items.Insert(2, "Channel 2");
    comboBox1.Items.Insert(3, "Channel 3");
    comboBox1.SelectedIndex = 0;
}

private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
   if(comboBox1.SelectedIndex == 1)
   {
      ClassGlobal.roomService = 1;
      MessageBox.Show("You selected Service CH 1");
      if (rbtn_SS.Checked == true || rbtnMS.Checked == true)
      {
         int group = 1;
         GoOut(group);
      }
      if (rbtnSM.Checked == true || rbtnMM.Checked == true)
      {
         int group = 2;
         GoOut(group);
      }
   }
   if(comboBox1.SelectedIndex == 2)
   {
      ClassGlobal.roomService = 2;
      MessageBox.Show("You selected Service CH 2");
      if (rbtnSM.Checked == true || rbtnMM.Checked == true)
      {
         int group = 2;
         GoOut(group);
```

```
            }
        }
        if (comboBox1.SelectedIndex == 3)
        {
            ClassGlobal.roomService = 3;
            MessageBox.Show("You selected Service CH 3");
            if (rbtnSM.Checked == true || rbtnMM.Checked == true)
            {
                int group = 2;
                GoOut(group);
            }
        }
    }

    private void GoOut(int Group)
    {
        if (Group == 1)
        {
            ClassGlobal.ServiceCH = Convert.ToDouble(comboBox1.SelectedIndex.ToString());
            frmSServer frmSServer = new frmSServer();
            frmSServer.Show();
        }
        if (Group == 2)
        {
            ClassGlobal.ServiceCH = Convert.ToDouble(comboBox1.SelectedIndex.ToString());
            frmMServer frmMServer = new frmMServer();
            frmMServer.Show();
        }
    }

    private void btnExit_Click(object sender, EventArgs e)
    {
        if (MessageBox.Show("คุณต้องการออกจากโปรแกรม ใช่หรือไม่...", "คำยืนยัน",
MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
        {
            Application.Exit();
```

```
            }
            else
                return;
        }

        private void btnShowBoard_Click(object sender, EventArgs e)
        {

            if (ClassGlobal.showBoard == "close")
            {
                ClassGlobal.frmBoard.Show();
                ClassGlobal.showBoard = "open";
            }
            else if (ClassGlobal.showBoard == "open")
            {
                ClassGlobal.frmBoard.Hide();
                ClassGlobal.showBoard = "close";
            }
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Collections;

namespace project_waiting_time
{
    public partial class frmBoard : Form
    {
        public frmBoard()
        {
            InitializeComponent();
            timer1.Enabled = true;
        }

        private void frmBoard_Load(object sender, EventArgs e)
        {
            lbNextQ.Text = ClassGlobal.nextQ.ToString();
            ClassGlobal.Load_StatusData();
            if (ClassGlobal.arrQValues[0] == "0" || ClassGlobal.arrQValues[0] == null ||
ClassGlobal.arrQValues[0] == "")
            {
                txtR1.Text = "ว่าง";
                txtR1.BackColor = Color.LawnGreen;
            }
            if (ClassGlobal.arrQValues[1] == "0" || ClassGlobal.arrQValues[1] == null ||
ClassGlobal.arrQValues[1] == "")
            {
                txtR2.Text = "ว่าง";
                txtR2.BackColor = Color.LawnGreen;
```

```
        }
        if (ClassGlobal.arrQValues[2] == "0" || ClassGlobal.arrQValues[2] == null ||
ClassGlobal.arrQValues[2] == "")
        {
            txtR3.Text = "ว่าง";
            txtR3.BackColor = Color.LawnGreen;
        }
    }

    private void timer1_Tick(object sender, EventArgs e)
    {
        ClassGlobal.Load_StatusData();
        if (ClassGlobal.arrQValues[0] == "0" || ClassGlobal.arrQValues[0] == null ||
ClassGlobal.arrQValues[0] == "")
        {
            txtR1.Text = "ว่าง";
            txtR1.BackColor = Color.LawnGreen;
        }
        else
        {
            txtR1.Text = ClassGlobal.arrQValues[0];
            txtR1.BackColor = Color.Red;
        }
        if (ClassGlobal.arrQValues[1] == "0" || ClassGlobal.arrQValues[1] == null ||
ClassGlobal.arrQValues[1] == "")
        {
            txtR2.Text = "ว่าง";
            txtR2.BackColor = Color.LawnGreen;
        }
        else
        {
            txtR2.Text = ClassGlobal.arrQValues[1];
            txtR2.BackColor = Color.Red;
        }
        if (ClassGlobal.arrQValues[2] == "0" || ClassGlobal.arrQValues[2] == null ||
ClassGlobal.arrQValues[2] == "")
```

```
            {
                txtR3.Text = "ว่าง";
                txtR3.BackColor = Color.LawnGreen;
            }
            else
            {
                txtR3.Text = ClassGlobal.arrQValues[2];
                txtR3.BackColor = Color.Red;
            }

            lbNextQ.Text = ClassGlobal.nextQ.ToString();
        }

        private void btnByPass_Click(object sender, EventArgs e)
        {
            ClassGlobal.Remove_Queue();
            if (ClassGlobal.MessageRemoveQ == "Yes")
                lbNextQ.Text = ClassGlobal.nextQ.ToString();
            else
                MessageBox.Show("ไม่มีคิวรอถัดไป !");
        }

        private void txtR1_TextChanged(object sender, EventArgs e)
        {

        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Collections;

namespace project_waiting_time
{
    public partial class frmFinalServices : Form
    {
        int minutes, seconds, queueNumber, roomNumber;
        string TimeNow = DateTime.Now.ToShortTimeString().Replace(':', '.');
        string DateNow = DateTime.Now.ToShortDateString().Replace('/', '_');

        public frmFinalServices(int queueNumber_Service, int room)
        {
            InitializeComponent();
            queueNumber = queueNumber_Service;
            roomNumber = room;
            lbRoomNumber.Text = roomNumber.ToString();
        }

        private void frmFinalServices_Load(object sender, EventArgs e)
        {
            lbQnumber.Text = queueNumber.ToString();
            minutes = int.Parse("00");
            seconds = int.Parse("00");
            timer1.Enabled = true;
        }

        private void timer1_Tick(object sender, EventArgs e)
```

```
{
    if (seconds >= 59)
    {
        seconds = 00;
        if (minutes == 59)
        {
            minutes = 00;
        }
        else
        {
            minutes += 1;
        }
    }
    else
        seconds += 1;
    btnMin.Text = minutes.ToString();
    btnSec.Text = seconds.ToString();
}

private void btnClose_Click(object sender, EventArgs e)
{
    if (!File.Exists(ClassGlobal.pathModel + @"\เวลาออก\" + DateNow + ".txt"))
    {
        StreamWriter SW;
        SW = File.CreateText(ClassGlobal.pathModel + @"\เวลาออก\" + DateNow + ".txt");
        SW.Close();
        string path = ClassGlobal.pathModel + @"\เวลาออก\" + DateNow + ".txt";
        AppendToFile(path);
    }
    else
    {
        string path = ClassGlobal.pathModel + @"\เวลาออก\" + DateNow + ".txt";
        AppendToFile(path);
    }
    ClassGlobal.arrQValues[roomNumber - 1] = "0";
    ClassGlobal.Save_StatusData();
```

```
            this.Close();
        }

        private void AppendToFile(string path)
        {
            StreamWriter SW;
            SW = File.AppendText(path);
            SW.WriteLine(TimeNow);
            SW.Close();
        }

        private void btnMin_Click(object sender, EventArgs e)
        {

        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Collections;

namespace project_waiting_time
{
    public partial class frmMServer : Form
    {
        string DateNow = DateTime.Now.ToShortDateString().Replace('/', '_');
        int amountQueue = 0;
        int queueNumber_Wait = 0;
        int queueNumber_Service = 0;
        string [] queueStatus = new string [2];
        int roomNumber = 0;
        double ServTime = 0;

        public frmMServer()
        {
            InitializeComponent();
            btnGetServiceR1.Visible = false;
            btnGetServiceR2.Visible = false;
            btnGetServiceR3.Visible = false;
            frmMServer2 frmMServer2 = new frmMServer2();
            frmMServer2.Show();
        }

        private void Form2_Load(object sender, EventArgs e)
        {
            timer1.Start();
```

```
            timer2.Enabled = false ;
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            lbTimer.Text = DateTime.Now.ToLongTimeString();
            lbDate.Text = DateTime.Now.ToLongDateString();
        }

        private void btnGetQueue_Click(object sender, EventArgs e)
        {
            ClassGlobal.enterTime =
 Convert.ToDouble(Convert.ToDouble(DateTime.Now.ToShortTimeString().Replace(':', '.')));
            ClassCalculate.multiServer_calculate();
            string path = @"D:\ProjectFinal\Data\Queue\1\" + DateNow + ".txt";
            queueStatus[0] = "queueWait";
            queueStatus[1] = "SaveEnter";
            AppendToFile(path);
            path = ClassGlobal.pathModel + @"\เวลามา\" + DateNow + ".txt";
            AppendToFile(path);
            frmWaiting frmWaiting = new frmWaiting(this.amountQueue, this.queueNumber_Wait);
            frmWaiting.Show();
        }

        private void btnGetServiceR1_Click(object sender, EventArgs e)
        {
/*          ServTime =
 Convert.ToDouble(Convert.ToDouble(DateTime.Now.ToShortTimeString().Replace(':', '.')));
            roomNumber = 1;
            GetServiceMS();
            ClassGlobal.Save_StatusData();
            ClassGlobal.Show_NextQueue();
            //Enter to services form (FinalServices)
            frmFinalServices frmFinalServices = new
 frmFinalServices(this.queueNumber_Service,this.roomNumber);
            frmFinalServices.Show();
```

```
*/      }

        private void btnGetServiceR2_Click(object sender, EventArgs e)
        {
/*          ServTime =
Convert.ToDouble(Convert.ToDouble(DateTime.Now.ToShortTimeString().Replace(':', '.')));
            ClassCalculate.multiServer_calculate();
            roomNumber = 2;
            GetServiceMS();
            ClassGlobal.Save_StatusData();
            ClassGlobal.Show_NextQueue();
            //Enter to services form (FinalServices)
            frmFinalServices frmFinalServices = new frmFinalServices(this.queueNumber_Service,
this.roomNumber);
            frmFinalServices.Show();
*/      }

        private void btnGetServiceR3_Click(object sender, EventArgs e)
        {
/*          ServTime =
Convert.ToDouble(Convert.ToDouble(DateTime.Now.ToShortTimeString().Replace(':', '.')));
            ClassCalculate.multiServer_calculate();
            roomNumber = 3;
            GetServiceMS();
            ClassGlobal.Save_StatusData();
            ClassGlobal.Show_NextQueue();
            //Enter to services form (FinalServices)
            frmFinalServices frmFinalServices = new frmFinalServices(this.queueNumber_Service,
this.roomNumber);
            frmFinalServices.Show();
*/      }

        private void GetServiceMS()
        {
/*          DateTime dtToday = new DateTime();
            dtToday = DateTime.Today.Date;//Current day
```

```
        string Today = String.Format("{0:d_M_yyyy}", dtToday);
        string path = @"D:\ProjectFinal\Data\Queue\2\" + Today + ".txt";
        queueStatus[0] = "queueServ";
        queueStatus[1] = "SaveServ";
        AppendToFile(path);
        path = ClassGlobal.pathModel + @"\เวลาเข้าบริการ\" + DateNow + ".txt";
        AppendToFile(path);
*/      }

    private void AppendToFile(string path)
    {
        if (queueStatus[0] == "queueWait" || queueStatus[0] == "queueServ")
        {
            int maxqueueNumber_Wait = 0, maxqueueNumber_Service = 0;
            ArrayList arrAppend1 = new ArrayList();
            ArrayList arrAppend2 = new ArrayList();
            arrAppend1.Clear();
            arrAppend2.Clear();
            string pathQ_In = @"D:\ProjectFinal\Data\Queue\1\" + DateNow + ".txt";
            string pathQ_Out = @"D:\ProjectFinal\Data\Queue\2\" + DateNow + ".txt";

            if (!File.Exists(pathQ_In))
            {
                StreamWriter SW;
                SW = File.CreateText(pathQ_In);
                SW.Close();
            }
            if (!File.Exists(pathQ_Out))
            {
                StreamWriter SW;
                SW = File.CreateText(pathQ_Out);
                SW.Close();
            }
            StreamReader objReaderAppend = new StreamReader(pathQ_In);
            string tempReadAppend = "";
            while ((tempReadAppend = objReaderAppend.ReadLine()) != null)
```

```
{
    arrAppend1.Add(int.Parse(tempReadAppend));
    maxqueueNumber_Wait = Convert.ToInt32(tempReadAppend);
}
objReaderAppend.Close();
objReaderAppend = new StreamReader(pathQ_Out);
while ((tempReadAppend = objReaderAppend.ReadLine()) != null)
{
    arrAppend2.Add(int.Parse(tempReadAppend));
    maxqueueNumber_Service = Convert.ToInt32(tempReadAppend);
}
objReaderAppend.Close();
if (queueStatus[0] == "queueServ")
{
   for (int i = 0; i <= arrAppend1.Count; i++)
   {
      if (arrAppend1.Count > arrAppend2.Count)
      {
         if (Convert.ToInt32(arrAppend1[i]) == maxqueueNumber_Service)
         {
            maxqueueNumber_Service = int.Parse(arrAppend1[i + 1].ToString());
            break;
         }
         if (Convert.ToInt32(arrAppend1[i]) > maxqueueNumber_Service)
         {
            maxqueueNumber_Service = int.Parse(arrAppend1[i].ToString());
            break;
         }
      }
      else
         maxqueueNumber_Service += 1;
}
queueNumber_Wait = maxqueueNumber_Wait + 1;
queueNumber_Service = maxqueueNumber_Service;
if (roomNumber == 1) ClassGlobal.arrQValues[0] = queueNumber_Service.ToString();
if (roomNumber == 2) ClassGlobal.arrQValues[1] = queueNumber_Service.ToString();
```

```
        if (roomNumber == 3) ClassGlobal.arrQValues[2] = queueNumber_Service.ToString();
        if (arrAppend1.Count <= arrAppend2.Count && queueStatus[0] == "queueServ")
        {
            StreamWriter swNotGetQueue;
            swNotGetQueue = File.AppendText(pathQ_In);
            swNotGetQueue.WriteLine(queueNumber_Service);
            swNotGetQueue.Close();
            arrAppend1.Add(queueNumber_Service);
        }
        StreamWriter swUpdate;
        swUpdate = File.AppendText(path);
        if (queueStatus[0] == "queueWait")
        {
            swUpdate.WriteLine(queueNumber_Wait);
            arrAppend1.Add(queueNumber_Wait);
        }
        if (queueStatus[0] == "queueServ")
        {
            swUpdate.WriteLine(queueNumber_Service);
            arrAppend2.Add(queueNumber_Service);
        }
        swUpdate.Close();
        if (queueStatus[0] == "queueWait")
            amountQueue = arrAppend1.Count - arrAppend2.Count;
        queueStatus[0] = "";
    }
    else
    {
        if (!File.Exists(path))
        {
            StreamWriter SW;
            SW = File.CreateText(path);
            SW.Close();
        }
        if (!File.Exists(path))
        {
```

```
                StreamWriter SW;
                SW = File.CreateText(path);
                SW.Close();
            }
            StreamWriter swWriteTime;
            swWriteTime = File.AppendText(path);
            if (queueStatus[1] == "SaveEnter") swWriteTime.WriteLine(ClassGlobal.enterTime);
            if (queueStatus[1] == "SaveServ") swWriteTime.WriteLine(ServTime);
            swWriteTime.Close();
            queueStatus[1] = "";
        }
    }
    private void frmMServer_FormClosing(object sender, FormClosingEventArgs e)
    {
        if (MessageBox.Show("คุณต้องการออกจากหน้านี้ ใช่หรือไม่...", "คำยืนยัน", MessageBoxButtons.YesNo,
MessageBoxIcon.Question) == DialogResult.Yes)
        {
            e.Cancel = false;
        }
        else
        {
            e.Cancel = true;
    }
    private void timer2_Tick(object sender, EventArgs e)
    {
        if (ClassGlobal.arrQValues[0] == "0" || ClassGlobal.arrQValues[0] == null ||
ClassGlobal.arrQValues[0] == "" )
            btnGetServiceR1.Visible = true;
        else
            btnGetServiceR1.Visible = false;
        if (ClassGlobal.arrQValues[1] == "0" || ClassGlobal.arrQValues[1] == null ||
ClassGlobal.arrQValues[1] == "" && (ClassGlobal.roomService != 1))
            btnGetServiceR2.Visible = true;
        else
            btnGetServiceR2.Visible = false;
```

```
            if (ClassGlobal.arrQValues[2] == "0" || ClassGlobal.arrQValues[2] == null ||
ClassGlobal.arrQValues[2] == "" && (ClassGlobal.roomService != 1 && ClassGlobal.roomService != 2))
                btnGetServiceR3.Visible = true;
            else
                btnGetServiceR3.Visible = false;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Collections;

namespace project_waiting_time
{
    public partial class frmMServer2 : Form
    {
        string DateNow = DateTime.Now.ToShortDateString().Replace('/', '_');
        int amountQueue = 0;
        int queueNumber_Wait = 0;
        int queueNumber_Service = 0;
        string [] queueStatus = new string [2];
        int roomNumber = 0;
        double ServTime = 0;

        public frmMServer2()
        {
            InitializeComponent();
            btnGetQueue_2.Visible = false;
            btnGetServiceR1_2.Visible = false;
            btnGetServiceR2_2.Visible = false;
            btnGetServiceR3_2.Visible = false;
        }

        private void Form2_Load(object sender, EventArgs e)
        {
            timer1_2.Start();
            timer2_2.Enabled = true ;
```

```
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            lbTimer_2.Text = DateTime.Now.ToLongTimeString();
            lbDate_2.Text = DateTime.Now.ToLongDateString();
        }

        private void btnGetQueue_2_Click(object sender, EventArgs e)
        {
/*          ClassGlobal.enterTime =
Convert.ToDouble(Convert.ToDouble(DateTime.Now.ToShortTimeString().Replace(':', '.')));
            ClassCalculate.multiServer_calculate();
            string path = @"D:\ProjectFinal\Data\Queue\1\" + DateNow + ".txt";
            queueStatus[0] = "queueWait";
            queueStatus[1] = "SaveEnter";
            AppendToFile(path);
            path = ClassGlobal.pathModel + @"\เวลามา\" + DateNow + ".txt";
            AppendToFile(path);
            frmWaiting frmWaiting = new frmWaiting(this.amountQueue, this.queueNumber_Wait);
            frmWaiting.Show();
*/      }

        private void btnGetServiceR1_2_Click(object sender, EventArgs e)
        {
            ServTime =
Convert.ToDouble(Convert.ToDouble(DateTime.Now.ToShortTimeString().Replace(':', '.')));
            roomNumber = 1;
            GetServiceMS();
            ClassGlobal.Save_StatusData();
            ClassGlobal.Show_NextQueue();
            frmFinalServices frmFinalServices = new
frmFinalServices(this.queueNumber_Service,this.roomNumber);
            frmFinalServices.Show();
        }
```

```
        private void btnGetServiceR2_2_Click(object sender, EventArgs e)
        {
            ServTime =
 Convert.ToDouble(Convert.ToDouble(DateTime.Now.ToShortTimeString().Replace(':', '.')));
            ClassCalculate.multiServer_calculate();
            roomNumber = 2;
            GetServiceMS();
            ClassGlobal.Save_StatusData();
            ClassGlobal.Show_NextQueue();
            frmFinalServices frmFinalServices = new frmFinalServices(this.queueNumber_Service,
 this.roomNumber);
            frmFinalServices.Show();
        }

        private void btnGetServiceR3_2_Click(object sender, EventArgs e)
        {
            ServTime =
 Convert.ToDouble(Convert.ToDouble(DateTime.Now.ToShortTimeString().Replace(':', '.')));
            ClassCalculate.multiServer_calculate();
            roomNumber = 3;
            GetServiceMS();
            ClassGlobal.Save_StatusData();
            ClassGlobal.Show_NextQueue();
            frmFinalServices frmFinalServices = new frmFinalServices(this.queueNumber_Service,
 this.roomNumber);
            frmFinalServices.Show();
        }

        private void GetServiceMS()
        {
            DateTime dtToday = new DateTime();
            dtToday = DateTime.Today.Date;
            string Today = String.Format("{0:d_M_yyyy}", dtToday);
            string path = @"D:\ProjectFinal\Data\Queue\2\" + Today + ".txt";
            queueStatus[0] = "queueServ";
            queueStatus[1] = "SaveServ";
```

```
    AppendToFile(path);
    path = ClassGlobal.pathModel + @"\เวลาเข้าบริการ\" + DateNow + ".txt";
    AppendToFile(path);
}

private void AppendToFile(string path)
{
    if (queueStatus[0] == "queueWait" || queueStatus[0] == "queueServ")
    {
        int maxqueueNumber_Wait = 0, maxqueueNumber_Service = 0;
        ArrayList arrAppend1 = new ArrayList();
        ArrayList arrAppend2 = new ArrayList();
        arrAppend1.Clear();
        arrAppend2.Clear();
        string pathQ_In = @"D:\ProjectFinal\Data\Queue\1\" + DateNow + ".txt";
        string pathQ_Out = @"D:\ProjectFinal\Data\Queue\2\" + DateNow + ".txt";

        if (!File.Exists(pathQ_In))
        {
            StreamWriter SW;
            SW = File.CreateText(pathQ_In);
            SW.Close();
        }
        if (!File.Exists(pathQ_Out))
        {
            StreamWriter SW;
            SW = File.CreateText(pathQ_Out);
            SW.Close();
        }
        StreamReader objReaderAppend = new StreamReader(pathQ_In);
        string tempReadAppend = "";
        while ((tempReadAppend = objReaderAppend.ReadLine()) != null)
        {
            arrAppend1.Add(int.Parse(tempReadAppend));
            maxqueueNumber_Wait = Convert.ToInt32(tempReadAppend);
        }
```

```
objReaderAppend.Close();
objReaderAppend = new StreamReader(pathQ_Out);
while ((tempReadAppend = objReaderAppend.ReadLine()) != null)
{
    arrAppend2.Add(int.Parse(tempReadAppend));
    maxqueueNumber_Service = Convert.ToInt32(tempReadAppend);
}
objReaderAppend.Close();
if (queueStatus[0] == "queueServ")
{
    for (int i = 0; i <= arrAppend1.Count; i++)
    {
        if (arrAppend1.Count > arrAppend2.Count)
        {
            if (Convert.ToInt32(arrAppend1[i]) == maxqueueNumber_Service)
            {
                maxqueueNumber_Service = int.Parse(arrAppend1[i + 1].ToString());
                break;
            }
            if (Convert.ToInt32(arrAppend1[i]) > maxqueueNumber_Service)
            {
                maxqueueNumber_Service = int.Parse(arrAppend1[i].ToString());
                break;
            }
        }
        else
            maxqueueNumber_Service += 1;
    }
}
queueNumber_Wait = maxqueueNumber_Wait + 1;
queueNumber_Service = maxqueueNumber_Service;
if (roomNumber == 1) ClassGlobal.arrQValues[0] = queueNumber_Service.ToString();
if (roomNumber == 2) ClassGlobal.arrQValues[1] = queueNumber_Service.ToString();
if (roomNumber == 3) ClassGlobal.arrQValues[2] = queueNumber_Service.ToString();
if (arrAppend1.Count <= arrAppend2.Count && queueStatus[0] == "queueServ")
{
```

```
            StreamWriter swNotGetQueue;
            swNotGetQueue = File.AppendText(pathQ_In);
            swNotGetQueue.WriteLine(queueNumber_Service);
            swNotGetQueue.Close();
            arrAppend1.Add(queueNumber_Service);
        }
        StreamWriter swUpdate;
        swUpdate = File.AppendText(path);
        if (queueStatus[0] == "queueWait")
        {
            swUpdate.WriteLine(queueNumber_Wait);
            arrAppend1.Add(queueNumber_Wait);
        }
        if (queueStatus[0] == "queueServ")
        {
            swUpdate.WriteLine(queueNumber_Service);
            arrAppend2.Add(queueNumber_Service);
        }
        swUpdate.Close();
        if (queueStatus[0] == "queueWait")
            amountQueue = arrAppend1.Count - arrAppend2.Count;
        queueStatus[0] = "";
    }
    else
    {
        if (!File.Exists(path))
        {
            StreamWriter SW;
            SW = File.CreateText(path);
            SW.Close();
        }
        if (!File.Exists(path))
        {
            StreamWriter SW;
            SW = File.CreateText(path);
            SW.Close();
```

```
            }
            StreamWriter swWriteTime;
            swWriteTime = File.AppendText(path);
            if (queueStatus[1] == "SaveEnter") swWriteTime.WriteLine(ClassGlobal.enterTime);
            if (queueStatus[1] == "SaveServ") swWriteTime.WriteLine(ServTime);
            swWriteTime.Close();
            queueStatus[1] = "";
        }
    }

    private void frmMServer_FormClosing(object sender, FormClosingEventArgs e)
    {
        if (MessageBox.Show("คุณต้องการออกจากหน้านี้ ใช่หรือไม่...", "คำยืนยัน", MessageBoxButtons.YesNo,
MessageBoxIcon.Question) == DialogResult.Yes)
        {
            e.Cancel = false;
        }
        else
        {
            e.Cancel = true;
        }
    }

    private void timer2_Tick(object sender, EventArgs e)
    {
        if (ClassGlobal.arrQValues[0] == "0" || ClassGlobal.arrQValues[0] == null ||
ClassGlobal.arrQValues[0] == "" )
            btnGetServiceR1_2.Visible = true;
        else
            btnGetServiceR1_2.Visible = false;
        if (ClassGlobal.arrQValues[1] == "0" || ClassGlobal.arrQValues[1] == null ||
ClassGlobal.arrQValues[1] == "" && (ClassGlobal.roomService != 1))
            btnGetServiceR2_2.Visible = true;
        else
            btnGetServiceR2_2.Visible = false;
```

```
            if (ClassGlobal.arrQValues[2] == "0" || ClassGlobal.arrQValues[2] == null ||
ClassGlobal.arrQValues[2] == "" && (ClassGlobal.roomService != 1 && ClassGlobal.roomService != 2))
                btnGetServiceR3_2.Visible = true;
            else
                btnGetServiceR3_2.Visible = false;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Collections;

namespace project_waiting_time
{
    public partial class frmSServer : Form
    {
        string DateNow = DateTime.Now.ToShortDateString().Replace('/', '_');
        int amountQueue = 0;
        int queueNumber_Wait = 0;
        int queueNumber_Service = 0;
        string[] queueStatus = new string[2];
        int roomNumber = 1;
        double ServTime = 0;

        public frmSServer()
        {
            InitializeComponent();
            btnGetService.Visible = false;
            frmSServer2 frmSServer2 = new frmSServer2();
            frmSServer2.Show();
        }

        private void Form2_Load(object sender, EventArgs e)
        {
            timer1.Start();
        }
```

```
        private void timer1_Tick(object sender, EventArgs e)
        {
            lbTimer.Text = DateTime.Now.ToLongTimeString();
            lbDate.Text = DateTime.Now.ToLongDateString();
        }

        private void btnGetQueue_Click(object sender, EventArgs e)
        {
            ClassGlobal.enterTime = Convert.ToDouble(DateTime.Now.ToShortTimeString().Replace(':', '.'));
            ClassCalculate.singleServer_calculate();
            string path = @"D:\ProjectFinal\Data\Queue\1\" + DateNow + ".txt";
            queueStatus[0] = "queueWait";
            queueStatus[1] = "SaveEnter";
            AppendToFile(path);
            path = ClassGlobal.pathModel + @"\เวลามา\" + DateNow + ".txt";
            AppendToFile(path);
            frmWaiting frmWaiting = new frmWaiting(this.amountQueue, this.queueNumber_Wait);
            frmWaiting.Show();
        }

        private void btnGetService_Click(object sender, EventArgs e)
        {
/*          ServTime =
Convert.ToDouble(Convert.ToDouble(DateTime.Now.ToShortTimeString().Replace(':', '.')));
            roomNumber = 1;
            GetServiceSS();
            ClassGlobal.Save_StatusData();
            ClassGlobal.Show_NextQueue();
            //Enter to services form (FinalServices)
            frmFinalServices frmFinalServices = new frmFinalServices(this.queueNumber_Service,
this.roomNumber);
            frmFinalServices.Show();
*/      }
```



```
        private void GetServiceSS()
        {
```

```
/*          DateTime dtToday = new DateTime();
            dtToday = DateTime.Today.Date;//Current day
            string Today = String.Format("{0:d_M_yyyy}", dtToday);
            //string TimeNow = String.Format("{0:h.mm}", dtToday);
            string path = @"D:\ProjectFinal\Data\Queue\2\" + Today + ".txt";
            queueStatus[0] = "queueServ"; //status for manage queue number
            queueStatus[1] = "SaveServ"; //status for service time to file
            AppendToFile(path);
            path = ClassGlobal.pathModel + @"\เวลาเข้าบริการ\" + DateNow + ".txt";
            AppendToFile(path);
*/      }

        private void AppendToFile(string path)
        {
            if (queueStatus[0] == "queueWait" || queueStatus[0] == "queueServ")
            {
                int maxqueueNumber_Wait = 0, maxqueueNumber_Service = 0;
                ArrayList arrAppend1 = new ArrayList();
                ArrayList arrAppend2 = new ArrayList();
                arrAppend1.Clear();
                arrAppend2.Clear();
                string pathQ_In = @"D:\ProjectFinal\Data\Queue\1\" + DateNow + ".txt";
                string pathQ_Out = @"D:\ProjectFinal\Data\Queue\2\" + DateNow + ".txt";

                if (!File.Exists(pathQ_In))
                {
                    StreamWriter SW;
                    SW = File.CreateText(pathQ_In);
                    SW.Close();
                }
                if (!File.Exists(pathQ_Out))
                {
                    StreamWriter SW;
                    SW = File.CreateText(pathQ_Out);
                    SW.Close();
                }
```

```
StreamReader objReaderAppend = new StreamReader(pathQ_In);
string tempReadAppend = "";
while ((tempReadAppend = objReaderAppend.ReadLine()) != null)
{
    arrAppend1.Add(int.Parse(tempReadAppend));
    maxqueueNumber_Wait = Convert.ToInt32(tempReadAppend);
}
objReaderAppend.Close();
objReaderAppend = new StreamReader(pathQ_Out);
while ((tempReadAppend = objReaderAppend.ReadLine()) != null)
{
    arrAppend2.Add(int.Parse(tempReadAppend));
    maxqueueNumber_Service = Convert.ToInt32(tempReadAppend);
}
objReaderAppend.Close();
if (queueStatus[0] == "queueServ")
{
    for (int i = 0; i <= arrAppend1.Count; i++)
    {
        if (arrAppend1.Count > arrAppend2.Count)
        {
            if (Convert.ToInt32(arrAppend1[i]) == maxqueueNumber_Service)
            {
                maxqueueNumber_Service = int.Parse(arrAppend1[i + 1].ToString());
                break;
            }
            if (Convert.ToInt32(arrAppend1[i]) > maxqueueNumber_Service)
            {
                maxqueueNumber_Service = int.Parse(arrAppend1[i].ToString());
                break;
            }
        }
        else
            maxqueueNumber_Service += 1;
    }
}
```

```
        queueNumber_Wait = maxqueueNumber_Wait + 1;
        queueNumber_Service = maxqueueNumber_Service;
        if (roomNumber == 1) ClassGlobal.arrQValues[0] = queueNumber_Service.ToString();
        if (arrAppend1.Count <= arrAppend2.Count && queueStatus[0] == "queueServ")
        {
            StreamWriter swNotGetQueue;
            swNotGetQueue = File.AppendText(pathQ_In);
            swNotGetQueue.WriteLine(queueNumber_Service);
            swNotGetQueue.Close();
            arrAppend1.Add(queueNumber_Service);
        }
        StreamWriter swUpdate;
        swUpdate = File.AppendText(path);
        if (queueStatus[0] == "queueWait")
        {
            swUpdate.WriteLine(queueNumber_Wait);
            arrAppend1.Add(queueNumber_Wait);
        }
        if (queueStatus[0] == "queueServ")
        {
            swUpdate.WriteLine(queueNumber_Service);
            arrAppend2.Add(queueNumber_Service);
        }
        swUpdate.Close();
        if (queueStatus[0] == "queueWait")
            amountQueue = arrAppend1.Count - arrAppend2.Count;
        queueStatus[0] = "";
    }
    else
    {
        if (!File.Exists(path))
        {
            StreamWriter SW;
            SW = File.CreateText(path);
            SW.Close();
        }
```

```
            if (!File.Exists(path))
            {
                StreamWriter SW;
                SW = File.CreateText(path);
                SW.Close();
            }
            StreamWriter swWriteTime;
            swWriteTime = File.AppendText(path);
            if (queueStatus[1] == "SaveEnter") swWriteTime.WriteLine(ClassGlobal.enterTime);
            if (queueStatus[1] == "SaveServ") swWriteTime.WriteLine(ServTime);
            swWriteTime.Close();
        }
    }
    private void frmSServer_FormClosing(object sender, FormClosingEventArgs e)
    {
        if (MessageBox.Show("คุณต้องการกลับไปหน้าหลัก ใช่หรือไม่...", "คำยืนยัน",
MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
        {
            e.Cancel = false;
        }
        else
        {
            e.Cancel = true;
        }
    }
  }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Collections;

namespace project_waiting_time
{
    public partial class frmSServer2 : Form
    {
        string DateNow = DateTime.Now.ToShortDateString().Replace('/', '_');
        int amountQueue = 0;
        int queueNumber_Wait = 0;
        int queueNumber_Service = 0;
        string[] queueStatus = new string[2];
        int roomNumber = 1;
        double ServTime = 0;

        public frmSServer2()
        {
            InitializeComponent();
            btnGetQueue_2.Visible = false;
        }

        private void Form2_Load(object sender, EventArgs e)
        {
            timer1_2.Start();
        }
```



```
        private void timer1_2_Tick(object sender, EventArgs e)
        {
```

```
            lbTimer_2.Text = DateTime.Now.ToLongTimeString();
            lbDate_2.Text = DateTime.Now.ToLongDateString();
        }

        private void btnGetQueue_2_Click(object sender, EventArgs e)
        {
/*          ClassGlobal.enterTime = Convert.ToDouble(DateTime.Now.ToShortTimeString().Replace(':',
'.'));
            ClassCalculate.singleServer_calculate();
            string path = @"D:\ProjectFinal\Data\Queue\1\" + DateNow + ".txt";
            queueStatus[0] = "queueWait"; //status for manage queue number
            queueStatus[1] = "SaveEnter"; //status enter time to file
            AppendToFile(path);
            path = ClassGlobal.pathModel + @"\เวลามา\" + DateNow + ".txt";
            AppendToFile(path);
            frmWaiting frmWaiting = new frmWaiting(this.amountQueue, this.queueNumber_Wait);
//Create new object frmWaiting
            frmWaiting.Show();
*/      }

        private void btnGetService_2_Click(object sender, EventArgs e)
        {
            ServTime =
Convert.ToDouble(Convert.ToDouble(DateTime.Now.ToShortTimeString().Replace(':', '.')));
            roomNumber = 1;
            GetServiceSS();
            ClassGlobal.Save_StatusData();
            ClassGlobal.Show_NextQueue();
            frmFinalServices frmFinalServices = new frmFinalServices(this.queueNumber_Service,
this.roomNumber);
            frmFinalServices.Show();
        }

        private void GetServiceSS()
        {
            DateTime dtToday = new DateTime();
```

```
    dtToday = DateTime.Today.Date;
    string Today = String.Format("{0:d_M_yyyy}", dtToday);
    string path = @"D:\ProjectFinal\Data\Queue\2\" + Today + ".txt";
    queueStatus[0] = "queueServ";
    queueStatus[1] = "SaveServ";
    AppendToFile(path);
    path = ClassGlobal.pathModel + @"\เวลาเข้าบริการ\" + DateNow + ".txt";
    AppendToFile(path);
}

private void AppendToFile(string path)
{
    if (queueStatus[0] == "queueWait" || queueStatus[0] == "queueServ")
    {
        int maxqueueNumber_Wait = 0, maxqueueNumber_Service = 0;
        ArrayList arrAppend1 = new ArrayList();
        ArrayList arrAppend2 = new ArrayList();
        arrAppend1.Clear();
        arrAppend2.Clear();
        string pathQ_In = @"D:\ProjectFinal\Data\Queue\1\" + DateNow + ".txt";
        string pathQ_Out = @"D:\ProjectFinal\Data\Queue\2\" + DateNow + ".txt";

        if (!File.Exists(pathQ_In))
        {
            StreamWriter SW;
            SW = File.CreateText(pathQ_In);
            SW.Close();
        }
        if (!File.Exists(pathQ_Out))
        {
            StreamWriter SW;
            SW = File.CreateText(pathQ_Out);
            SW.Close();
        }
        StreamReader objReaderAppend = new StreamReader(pathQ_In);
        string tempReadAppend = "";
```

```
while ((tempReadAppend = objReaderAppend.ReadLine()) != null)
{
    arrAppend1.Add(int.Parse(tempReadAppend));
    maxqueueNumber_Wait = Convert.ToInt32(tempReadAppend);
}
objReaderAppend.Close();
objReaderAppend = new StreamReader(pathQ_Out);
while ((tempReadAppend = objReaderAppend.ReadLine()) != null)
{
    arrAppend2.Add(int.Parse(tempReadAppend));
    maxqueueNumber_Service = Convert.ToInt32(tempReadAppend);
}
objReaderAppend.Close();
if (queueStatus[0] == "queueServ")
{
   for (int i = 0; i <= arrAppend1.Count; i++)
   {
      if (arrAppend1.Count > arrAppend2.Count)
      {
         if (Convert.ToInt32(arrAppend1[i]) == maxqueueNumber_Service)
         {
            maxqueueNumber_Service = int.Parse(arrAppend1[i + 1].ToString());
            break;
         }
         if (Convert.ToInt32(arrAppend1[i]) > maxqueueNumber_Service)
         {
            maxqueueNumber_Service = int.Parse(arrAppend1[i].ToString());
            break;
         }
      }
      else
         maxqueueNumber_Service += 1;
   }
}
queueNumber_Wait = maxqueueNumber_Wait + 1;
queueNumber_Service = maxqueueNumber_Service;
```

```
        if (roomNumber == 1) ClassGlobal.arrQValues[0] = queueNumber_Service.ToString();
        if (arrAppend1.Count <= arrAppend2.Count && queueStatus[0] == "queueServ")
        {
            StreamWriter swNotGetQueue;
            swNotGetQueue = File.AppendText(pathQ_In);
            swNotGetQueue.WriteLine(queueNumber_Service);
            swNotGetQueue.Close();
            arrAppend1.Add(queueNumber_Service);
        }
        StreamWriter swUpdate;
        swUpdate = File.AppendText(path);
        if (queueStatus[0] == "queueWait")
        {
            swUpdate.WriteLine(queueNumber_Wait);
            arrAppend1.Add(queueNumber_Wait);
        }
        if (queueStatus[0] == "queueServ")
        {
            swUpdate.WriteLine(queueNumber_Service);
            arrAppend2.Add(queueNumber_Service);
        }
        swUpdate.Close();
        if (queueStatus[0] == "queueWait")
            amountQueue = arrAppend1.Count - arrAppend2.Count;
        queueStatus[0] = "";
    }
    else
    {
        if (!File.Exists(path))
        {
            StreamWriter SW;
            SW = File.CreateText(path);
            SW.Close();
        }
        if (!File.Exists(path))
        {
```

```
                StreamWriter SW;
                SW = File.CreateText(path);
                SW.Close();
            }
            StreamWriter swWriteTime;
            swWriteTime = File.AppendText(path);
            if (queueStatus[1] == "SaveEnter") swWriteTime.WriteLine(ClassGlobal.enterTime);
            if (queueStatus[1] == "SaveServ") swWriteTime.WriteLine(ServTime);
            swWriteTime.Close();
        }
    }

    private void frmSServer_FormClosing(object sender, FormClosingEventArgs e)
    {
        if (MessageBox.Show("คุณต้องการกลับไปหน้าหลัก ใช่หรือไม่...", "คำยืนยัน",
MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
        {
            e.Cancel = false;
        }
        else
        {
            e.Cancel = true;
    }
  }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Collections;

namespace project_waiting_time
{
    public partial class frmWaiting : Form
    {
        int minutes, seconds, amountQueue, queueNumber;
        double WqShow = 0;
        string DateNow = DateTime.Now.ToShortDateString().Replace('/', '_');
        public frmWaiting(int amountQ, int queueNumber_Wait)
        {
            InitializeComponent();
            WqShow = ClassGlobal.Wq;
            amountQueue = amountQ;
            queueNumber = queueNumber_Wait;
            ConvertTime();
        }

        private void frmWaiting_Load(object sender, EventArgs e)
        {
            lbQnumber.Text = queueNumber.ToString();
            lbQueueCount.Text = "จำนวนคิวที่กำลังรอ: " + amountQueue.ToString() + " คิว";
            timer1.Start();
            string timeText = WqShow.ToString("00.00");
            string []timeArr = timeText.Split('.');
            minutes = int.Parse(timeArr[0].ToString());
            seconds = int.Parse(timeArr[1].ToString());
```

```
    timer2.Enabled = false;
}

private void timer1_Tick(object sender, EventArgs e)
{
    lbTimer.Text = "ขณะนี้เวลา  " + DateTime.Now.ToLongTimeString();
}

private void ConvertTime()
{
    string chkTime = WqShow.ToString("F2");
    string[] strArr = null;
    strArr = chkTime.Split('.');
    int rsTime = Convert.ToInt32(strArr[1]);
    int modTime = rsTime % 59;
    int divieTime = rsTime / 59;
      strArr[1] = modTime.ToString();
      strArr[0] = (Convert.ToInt32(strArr[0]) + divieTime).ToString();
    btnMin.Text = strArr[0];
    btnSec.Text = strArr[1];
}

private void timer2_Tick(object sender, EventArgs e)
{
   if ((minutes != 0) || (seconds != 0))
   {
      if (seconds < 1)
      {
         seconds = 59;
         if (minutes == 0)
         {
            minutes = 59;
         }
         else
         {
            minutes -= 1;
```

```
            }
        }
        else
            seconds -= 1;
    }
    else
    {
        timer2.Enabled = false;
        MessageBox.Show("ได้เวลาเข้ารับบริการของคุณแล้วค่ะ\n กรุณาเตรียมตัวค่ะ");
    }
     btnMin.Text = minutes.ToString();
     btnSec.Text = seconds.ToString();
}

private void btnMin_Click(object sender, EventArgs e)
{

}
}
}
```