B6607012 นายธนัช ตั้งมั่น

ระบบหลัก ระบบบริการการศึกษาแก่นักศึกษา (REG)
ระบบย่อย ระบบคำร้อง

ระบบคำร้อง เป็นระบบย่อยที่ออกแบบมาเพื่ออำนวยความสะดวกให้กับนักศึกษา ในการดำเนินการยื่นคำร้องด้านการศึกษาได้อย่างสะดวก รวดเร็ว และเป็นระบบ โดยไม่ต้องพึ่งพาการส่งเอกสารฉบับจริงด้วย ระบบนี้ช่วยลดภาระด้านเอกสาร ลดการตกหล่นของเอกสาร ลดความซ้ำซ้อนในขั้นตอนการดำเนินการ และสามารถติดตามสถานะคำร้อง เมื่อนักศึกษาต้องการยื่นคำร้อง เช่น การลาพักการเรียน ขอเพิ่ม/ถอนวิชา หรือขอแก้ผลการเรียน นักศึกษาสามารถเข้าสู่ระบบเพื่อกรอกแบบฟอร์มคำร้องได้ทันที โดยระบุประเภทคำร้อง ที่ตรงกับเรื่องที่ต้องการดำเนินการ พร้อมกรอก รายละเอียดเหตุผล อย่างชัดเจนเพื่อใช้ประกอบการพิจารณา ระบบจะบันทึกวันที่ยื่นคำร้อง เพื่อเป็นข้อมูลอ้างอิง และสามารถแนบไฟล์แนบ เช่น หนังสือรับรองแพทย์ หรือเอกสารประกอบอื่นๆ เพิ่มเติมได้ในกรณีที่จำเป็น หลังจากยื่นคำร้องแล้ว ระบบจะแสดงสถานะคำร้องแบบเรียลไทม์ เช่น “รอดำเนินการ”, “อนุมัติ” หรือ “ไม่อนุมัติ” เพื่อให้นักศึกษาสามารถติดตามความคืบหน้าของคำร้องได้ตลอดเวลา และนักศึกษาสามารถยื่นคำร้องให้ผู้พิจารณา(อาจารย์ หรือเจ้าหน้าที่)ได้โดยตรงได้

User Story ระบบคำร้อง
ในบทบาทของ (As a) นักศึกษา
ฉันต้องการ (I want to) ยื่นคำร้องผ่านระบบออนไลน์ เช่น คำร้องลาพักการเรียน, ขอเพิ่ม/ถอนวิชา หรือแก้ไขผลการเรียน
เพื่อ (So that) สามารถติดตามสถานะคำร้องได้ และไม่ต้องยื่นเอกสารซ้ำซ้อน

Output บนหน้าจอ

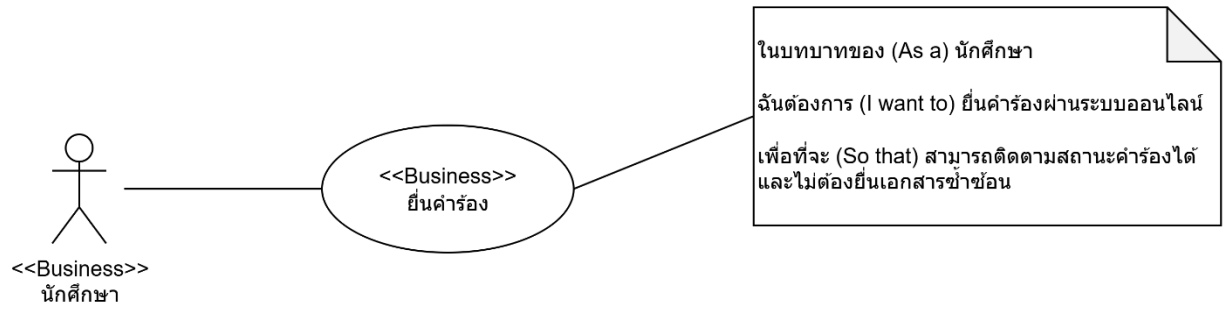
- นักศึกษาล็อกอินเข้าสู่ระบบ ⟶ เข้าสู่เมนู “ระบบคำร้อง” ⟶ กรอกข้อมูลประเภทคำร้อง ⟶ เลือกบุคคลที่จะยื่นคำร้องให้ ⟶ กรอกรายละเอียดคำร้อง ⟶ กดยืนยัน ⟶ ระบบแสดง “บันทึกการยื่นคำร้องแล้ว”

Output ของข้อมูล
ระบบรับข้อมูลการยื่นคำร้อง ⟶ บันทึกข้อมูลคำร้องใหม่ในฐานข้อมูล ⟶ กำหนดสถานะเป็น “รอดำเนินการ” ⟶ บันทึกวัน-เวลา และรหัสผู้ยื่นคำร้อง

คำนามที่อาจจะกลายมาเป็นตารางในฐานข้อมูลของระบบ

คำนาม	เหตุผล (เกี่ยวข้องกับ Use Case หรือไม่)
นักศึกษา (Student)	เกี่ยวข้องโดยตรง เพราะเป็นผู้ยื่นคำร้อง
ประเภทคำร้อง (Report Type)	เกี่ยวข้องโดยตรง เพราะใช้จำแนกประเภทของคำร้อง
รายละเอียดเหตุผล (Description)	เกี่ยวข้องโดยตรง เพราะแสดงเนื้อหาที่นักศึกษาพิมพ์เพื่ออธิบายเหตุผล
วันที่ยื่นคำร้อง (Submission Date)	เกี่ยวข้องโดยตรง เพราะใช้ระบุวันที่ยื่นคำร้องเพื่อบันทึกและอ้างอิง
สถานะคำร้อง (Report Status)	เกี่ยวข้องโดยตรง เพราะใช้ระบุสถานะคำร้อง เช่น รอดดำเนินการ, อนุมัติ, ไม่อนุมัติ
ไฟล์แนบ (Attachment)	เกี่ยวข้องโดยอ้อม เพราะใช้แนบเอกสารที่ใช้สนับสนุน ยืนยัน เช่น ใบรับรองแพทย์
ผู้พิจารณา (Reviewer)	เกี่ยวข้องโดยตรง เพราะให้เจ้าหน้าที่หรืออาจารย์ตรวจสอบคำร้อง

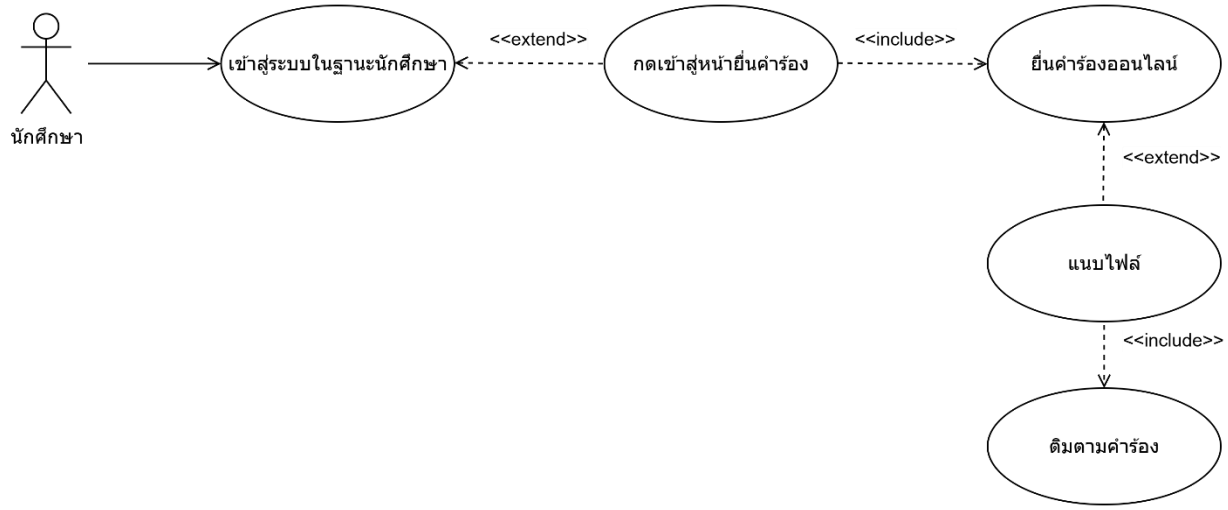
Business Use Case Diagram



Checklist: Business Use Case Diagram

- Business Actor มี<<Business>> กำกับ
- Business Actor เป็นชื่อบทบาท ไม่ใช่ชื่อบุคคล
- Business Use Case มี<<Business>> กำกับ
- Business Use Case ขึ้นต้นด้วยคำที่แสดงพฤติกรรม (คำกริยา)
- พฤติกรรมใน Business Use Case มาจาก User Story
- มี Note ที่แสดง User Story ใน Diagram

System Use Case Diagram



Checklist: System Use Case Diagram

- System Actor และ System Use Case ต้องไม่มีอะไรเชื่อมกัน
- เส้นโยงจาก System Actor ไปยัง System Use Case ต้องมีหัวลูกศรปลายเปิด
- System Actor ต้องเป็นบทบาทของคนที่ใช้งานจริงๆ
- System Use Case ต้องเป็นคำแสดงพฤติกรรม
- หากเป็นการรักษาความปลอดภัย ชื่อ System Use Case จะอยู่ในรูปแบบ "เข้าระบบในฐานะ <Actor>"
- การใช้<<extend>> เป็นเส้นที่หัวลูกศรปลายเปิด ชี้จาก System Use Case ไปยัง Use Case หลัก
- • การใช้<<include>> เป็นเส้นที่หัวลูกศรปลายเปิด ชี้จาก System Use Case ไปยัง System Use Case ที่ถูกเชื่อมต่อ

วิเคราะห์ Entity ที่เกี่ยวข้อง

นักศึกษา (Student)

- ชื่อ Entity: Student
- ข้อมูลที่ต้องจัดเก็บ
  - Student_id: เป็น Primary Key เก็บในรูปแบบ varchar ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
  - FirstName: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - LastName: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Citizen_id: เก็บในรูปแบบ char และไม่สามารถเป็นค่า Null ได้
  - Gender: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Email: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Phone: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้

STUDENT_ID VERCHAR NOT NULL, PK	**FIRSTNAME** VERCHAR NOT NULL	**LASTNAME** VERCHAR NOT NULL	**CITIZEN_ID** CHAR NOT NULL	**GENDER** VERCHAR NOT NULL	**EMAIL** VARCHAR NOT NULL	**PHONE** VERCHAR NOT NULL
B0000001	สมปอง	สองใจ	1302213456789	ชาย	Sompong@gmail.com	454-564
B0000002	สมหมาย	ดีใจ	1301234567890	ชาย	Sommai@gmail.com	594-412

ผู้ใช้ (User)

- ชื่อ Entity: User
- ข้อมูลที่ต้องจัดเก็บ
  - User_id: เป็น Primary Key เก็บในรูปแบบ varchar ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
  - Password: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้

- Role: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้

USER_ID VARCHAR NOT NULL, PK	PASSWORD VARCHAR NOT NULL	ROLE VARCHAR NOTNULL
B0000001	1302213456789	Student
B0000002	1301234567890	Student

ผู้พิจารณา (Reviewer)

- ชื่อ Entity: Reviewer
- ข้อมูลที่ต้องจัดเก็บ
  - Reviewer_id: เป็น Primary Key เก็บในรูปแบบของ varchar ที่ไม่สามารถซ้ำกันได้ และไม่สามารถเป็น Null ได้
  - User_id: เป็น Foreign Key เก็บในรูปแบบของ varchar ที่ไม่สามารถซ้ำกันได้ และไม่สามารถเป็น Null ได้

REVIEWER_ID VARCHAR NOT NULL, PK	USER_ID VARCHAR NOT NULL, PK
101	1
102	2

ความคิดเห็นของผู้พิจารณา (ReviewerComment)

- ชื่อ Entity: ReviewerComment
- ข้อมูลที่ต้องจัดเก็บ
  - Comment_id: เป็น Primary Key เก็บในรูปแบบของ varchar ที่ไม่สามารถซ้ำกันได้ และไม่สามารถเป็น Null ได้

- Report_id: เป็น Foreign Key เก็บในรูปแบบของ varchar ที่ไม่สามารถซ้ำกันได้ และไม่สามารถเป็น Null ได้
- Reviewer_id: เป็น Foreign Key เก็บในรูปแบบของ varchar ที่ไม่สามารถซ้ำกันได้ และไม่สามารถเป็น Null ได้
- CommentText: เก็บในรูปแบบของ text
- CommentDate: เก็บในรูปแบบของ datetime และไม่สามารถเป็นค่า Null ได้

COMMENT_ID VARCHAR NOT NULL, PK	REPORT_ID VARCHAR NOT NULL, FK	REVIEWER_ID VARCHAR NOT NULL, FK	COMMENTTEXT TEXT	COMMENTDATE DATETIME NOT NULL
1	1	101	นักศึกษามีเหตุผล ที่สมเหตุสมผล อาจารย์จึงอนุมัติ	2025-07-21 10:30:00
2	2	102	-	2025-07-22 10:30:20

คำร้อง (Report)

- ชื่อ Entity: Report
- ข้อมูลที่ต้องจัดเก็บ
  - Report_id: เป็น Primary Key เก็บในรูปแบบของ varchar ที่ไม่สามารถซ้ำกันได้ และไม่สามารถเป็น Null ได้
  - Student_id: เป็น Foreign Key เก็บในรูปแบบของ varchar ที่ไม่สามารถซ้ำกันได้ และไม่สามารถเป็น Null ได้
  - Reviewer_id: เป็น Foreign Key เก็บในรูปแบบของ varchar ที่ไม่สามารถซ้ำกันได้ และไม่สามารถเป็น Null ได้
  - ReportType_id: เป็น Foreign Key เก็บในรูปแบบของ varchar ที่ไม่สามารถซ้ำกันได้ และไม่สามารถเป็น Null ได้
  - Attchment_id: เป็น Foreign Key เก็บในรูปแบบของ varchar ที่ไม่สามารถซ้ำกันได้ และไม่สามารถเป็น Null ได้
  - Report_details: เก็บในรูปแบบของ text
  - Submission_date: เก็บในรูปแบบของ datetime และไม่สามารถเป็นค่า Null ได้

- status: เก็บในรูปแบบของ varchar และไม่สามารถเป็นค่า Null ได้

REPORT_ID VARCHAR NOT NULL, PK	**STUDENT_ID** VARCHAR NOT NULL, FK	**REVIEWER_ID** VARCHAR NOT NULL, FK	**REPORTTYPE_ID** VARCHAR NOT NULL, FK	**ATTCHMENT_ID** VARCHAR NOT NULL, FK	**REPORT_DETAILS** TEXT	**SUBMISSION_DATE** DATETIME NOT NULL	**STATUS** VARCHAR NOT NULL
1	0001	1	2	1	ขอเพิ่มหน่วยกิตเนื่องจากต้องการเรียนเพิ่มเพื่อสำเร็จการศึกษารวดเร็ว	2025-07-20 13:05:00	อนุมัติ
2	0002	2	1	3	ขอลาพักการเรียนเนื่องจากปัญหาสุขภาพ	2025-07-22 10:30:20	รอดำเนินการ

ประเภทคำร้อง (ReportType)

- ชื่อ Entity: ReportType
- ข้อมูลที่ต้องจัดเก็บ
  - ReportType_id: เป็น Primary Key เก็บในรูปแบบของ varchar ที่ไม่สามารถซ้ำกันได้และไม่สามารถเป็น Null ได้
  - ReportType_Name: เก็บในรูปแบบของ varchar และไม่สามารถเป็น Null ได้
  - Description: เก็บในรูปแบบของ text

REPORTTYPE_ID VARCHAR NOT NULL, PK	REPORTTYPE_NAME VARCHAR NOT NULL	DESCRIPTION TEXT
1	ลาพักการเรียน	สำหรับนักศึกษาที่มีเหตุจำเป็นต้องหยุดการเรียนชั่วคราว
2	ขอลงทะเบียนเรียนเพิ่ม(เกินหน่วยกิต)	สำหรับนักศึกษาที่มีความจำเป็นต้องการลงทะเบียนมากกว่าที่เกณฑ์กำหนดไว้

ไฟล์แนบ (Attachment)

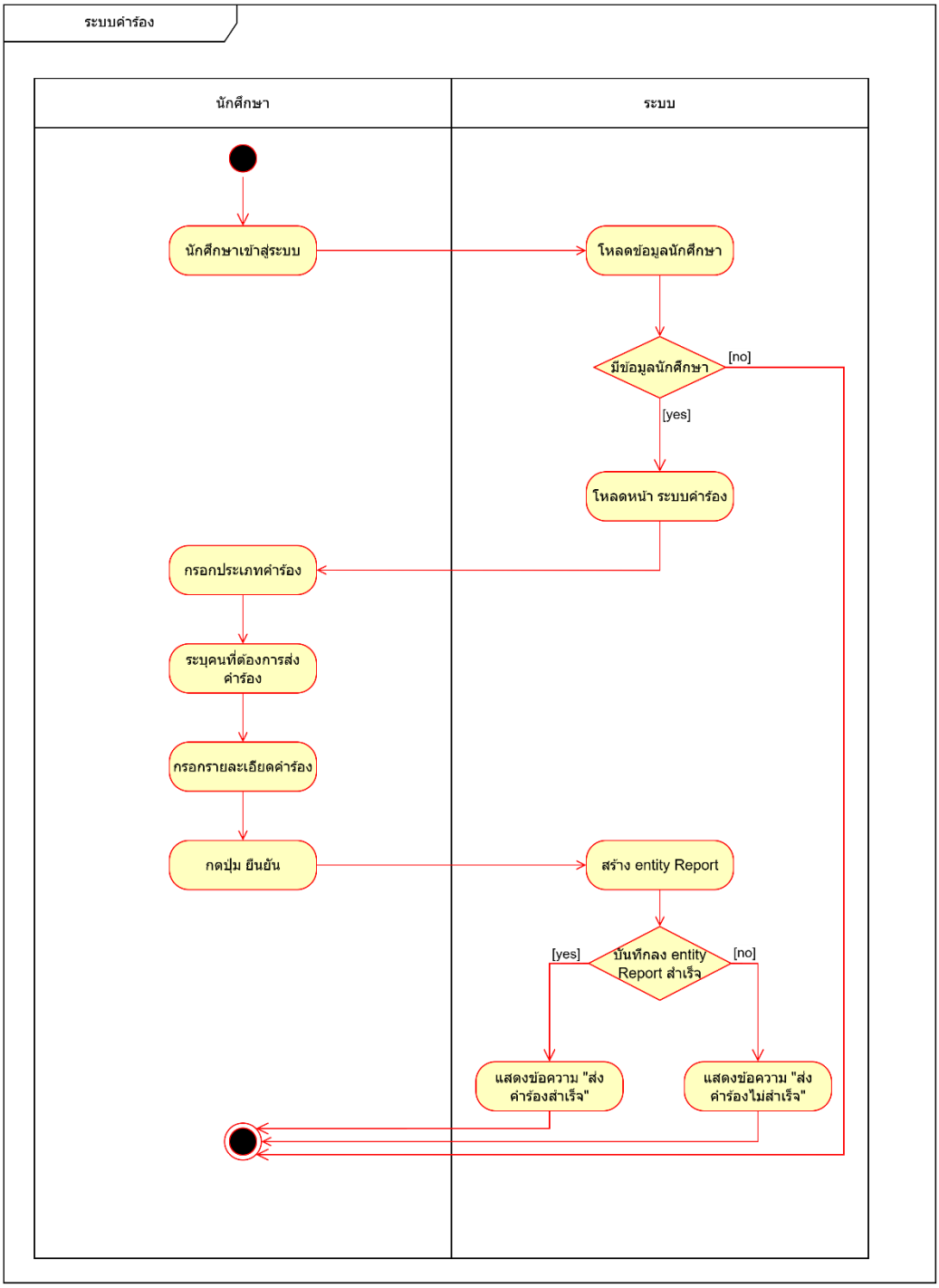
- ชื่อ Entity: Attachment
- ข้อมูลที่ต้องจัดเก็บ
  - Attachment_id: เป็น Primary Key เก็บในรูปแบบของ varchar ที่ไม่สามารถซ้ำกันได้ และไม่สามารถเป็น Null ได้
  - File_Name: เก็บในรูปแบบของ varchar และไม่สามารถเป็นค่า Null ได้
  - File_Path: เก็บในรูปแบบของ varchar และไม่สามารถเป็นค่า Null ได้
  - Uploaded_date: เก็บในรูปแบบของ datetime และไม่สามารถเป็นค่า Null ได้

ATTACHMENT_ID VARCHAR NOT NULL, PK	FILE_NAME VARCHAR NOT NULL	FILE_PATH VARCHAR NOT NULL	UPLOADED_DATE DATETIME NOT NULL
2	ใบรับรองแพทย์	/medical_certificate	2025-07-22 09:15:04
3	หนังสือขอลาพักการเรียน	/study_leave_request	2025-07-22 10:30:20

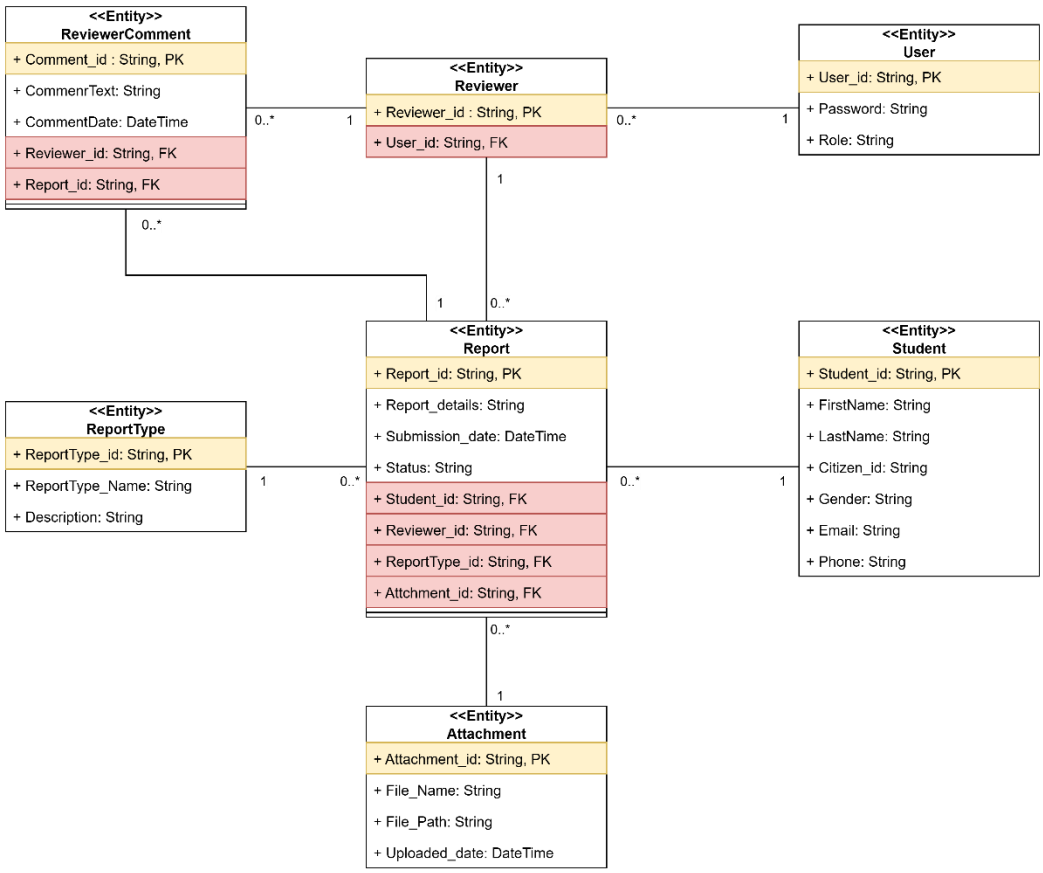
UI Design


ยินดีต้อนรับ | ธนัช ตั้งมั่น
ARCANATECH UNIVERSITY
ลงทะเบียนเรียน
วิชาที่เปิดสอน
ตารางเรียน
ผลการเรียน
คะแนน
ใบแจ้งยอดชำระ
ระเบียนประวัติ
อาจารย์
คำร้อง
แจ้งจบการศึกษา
หลักสูตร
เปลี่ยนรหัสผ่าน
ออกจากระบบ
หน้าหลัก
เลือกประเภทคำร้อง▼
อาจารย์ / เจ้าหน้าที่▼
กรอกรายละเอียดคำร้อง
กรอกคำร้อง
แนบไฟล์
ส่งคำร้อง

Activity Diagram



Class Diagram



Checklist for Class Diagram

- ต้องมี Tag <<Entity>> กำกับในส่วนของหัวตาราง
- ในรายละเอียดของ Field ต้องบอกได้ทั้งชื่อ Field และ Datatype ที่จะนำมาใช้งานในการพัฒนาได้จริง Primary Key จะต้องเติม PK ต่อท้าย Datatype
- Foreign Key จะต้องมีกรอบสีแดงกำกับ
- ในส่วนของ Relationship ระหว่าง Entity จะต้องวาดด้วยเส้นตรง และมีตัวเลข 1 และ 0..* ในการกำกับความสัมพันธ์

Backend

Entity – Report.go

```
package entity
import (
    "time"
    "gorm.io/gorm"
)
type Report struct {
    gorm.Model
    Report_id       string     `gorm:"primaryKey" json:"Report_id"`
    Report_details string     `json:"Report_details"`
    Submittion_date  time.Time `json:"ReportSubmission_date"`
    Status             string     `json:"ReportStatus"`

    Student_id uint
    Student     Students `gorm:"foreignKey:Student_id"`

    Reviewer_id uint
    Reviewer     Reviewer `gorm:"foreignKey:Reviewer_id"`

    ReportType_id uint
    ReportType     ReportType `gorm:"foreignKey:ReportType_id"`

    Attachment_id uint
    Attachment     Attachment `gorm:"foreignKey:Attachment_id"`
}
```

Entity – ReportType.go

```
package entity
import (
    "gorm.io/gorm"
)
type ReportType struct {
    gorm.Model
    ReportType_id    string `gorm:"primaryKey" json:"ReportType_id"`
    ReportType_Name string `json:"ReportType_Name"`
    Description        string `json:"ReportTypeDescription"`

    Reports []Report `gorm:"foreignKey:ReportType_id"`
}
```

Entity – Attachment.go

```
package entity
import (
    "time"
    "gorm.io/gorm"
)
type Attachment struct {
    gorm.Model
    Attachment_id string     `gorm:"primaryKey" json:"Attachment_id"`
    File_Name      string     `json:"Attachment_File_Name"`
    File_Path      string     `json:"Attachment_File_Path"`
    Uploaded_date time.Time `json:"Attachment_Uploaded_date"`

    Reports []Report `gorm:"foreignKey:Attachment_id"`
}
```

Entity – Reviewer.go

```
package entity
import (
    "time"
    "gorm.io/gorm"
)
type Reviewer struct {
    gorm.Model
    Reviewer_id        string     `gorm:"primaryKey" json:"Reviewer_id"`
    FirstName          string     `json:"Reviewer_FirstName"`
    LastName           string     `json:"Reviewer_LastName"`
    Email              string     `json:"Reviewer_Email"`
    Reviewer_comments string     `json:"Reviewer_comments"`
    Reviewer_date      time.Time `json:"Reviewer_date"`

    Reports []Report `gorm:"foreignKey:Reviewer_id"`
}
```

FRONTEND

Report.tsx

```
import React from 'react';
import { Layout, Select, Card, Button, Upload, Input } from 'antd';
import './report.css';

const { Header, Content, Footer } = Layout;
const wrapperStyle: React.CSSProperties = {
  borderRadius: 8,
  boxShadow: '0 4px 12px rgba(0,0,0,0.08)',
```

```
  width: '100%',
  minHeight: '100vh',
  display: 'flex',
  flexDirection: 'column',
  overflow: 'hidden',
};
const headerStyle: React.CSSProperties = {
  background: '#2e236c',
  color: 'white',
  textAlign: 'center',
  padding: 16,
  fontSize: 20,
  fontWeight: 'bold',
};
const contentStyle: React.CSSProperties = {
  background: '#f5f5f5',
  padding: 24,
  minHeight: 400,
  color: '#333',
  overflowY: 'auto',
};
const footerStyle: React.CSSProperties = {
  background: '#1890ff',
  color: 'white',
  textAlign: 'center',
  padding: 12,
};
const Report: React.FC = () => {
  return (
    <Layout style={wrapperStyle}>
      <Header style={headerStyle}>ระบบคำร้อง</Header>
      <Content style={contentStyle}>
        <div style={{ display: 'flex', gap: 100, flexWrap: 'wrap', justifyContent: 'center' }}>
          <Select
        className="report-select"
```

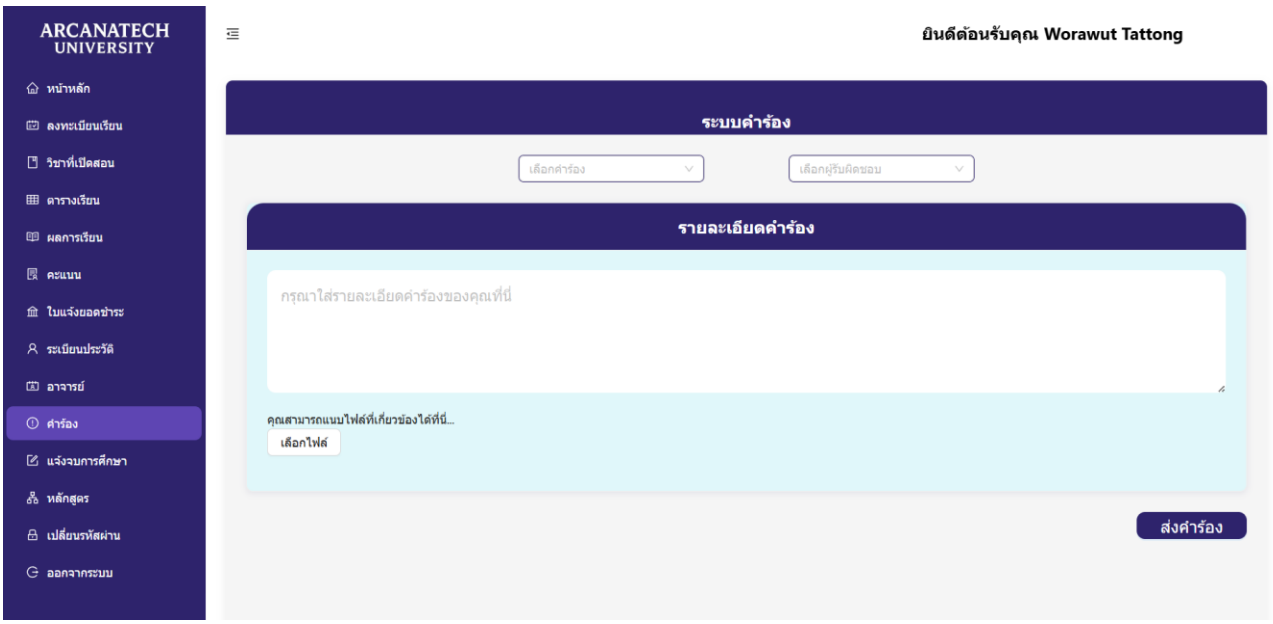
```
  style={{ width: 220 }}
  placeholder="เลือกคำร้อง"
  options={[
    { value: 'LeaveofAbsence', label: 'ลาพักการเรียน' },
    { value: 'AddCourses', label: 'ขอลงทะเบียนเรียนเพิ่ม' },
  ]}
    />
    <Select
  className="person-select"
  style={{ width: 220 }}
  placeholder="เลือกผู้รับผิดชอบ"
  options={[
    { value: 'teacher1', label: 'สมชาย ใจดี' },
    { value: 'teacher2', label: 'สมใจ ปองภพ' },
    { value: 'staff', label: 'เจ้าหน้าที่' },
  ]}
    />
  </div>
  <Card className="report-card" title="รายละเอียดคำร้อง" style={{ marginTop: 24 }}>
    <Input.TextArea
  rows={4}
  placeholder="กรุณาใส่รายละเอียดคำร้องของคุณที่นี่"
  style={{ marginBottom: 20 , backgroundColor: 'white'}}
    />
    <p>คุณสามารถแนบไฟล์ที่เกี่ยวข้องได้ที่นี่...</p>
    <Upload className="report-file-input" beforeUpload={() => false} multiple>
  <Button>เลือกไฟล์</Button>
    </Upload>
  </Card>
  <div style={{ display: 'flex', justifyContent: 'flex-end', marginTop: 24 }}>
    <Button type="primary" className="report-submit-button">
  ส่งคำร้อง
    </Button>
  </div>
</Content>
```

```
      <Footer style={footerStyle}>Footer © 2025</Footer>
    </Layout>
  );
};
export default Report;
```

Report.css

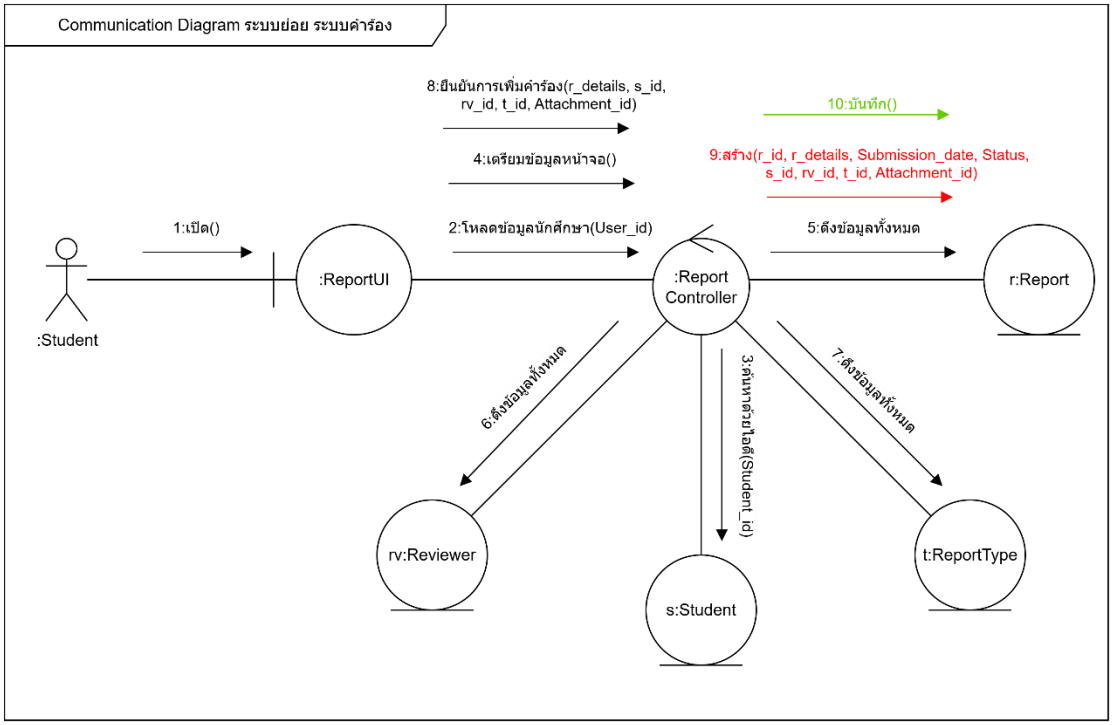
```
.report-select,
.person-select {
    display: flex;
    width: 240px;
    border: 1.5px solid #2e236c;
    border-radius: 6px;
    background: white;
    font-size: 16px;
}
.report-select:focus,
.person-select:focus {
    border-color: #5a4fcf;
    outline: none;
}
.report-card .ant-card-head {
    width: 100%;
    background-color: #2e236c;
    color: #fff;
    font-size: 20px;
    text-align: center;
    font-style: normal;
    display: flex;
    justify-content: center;
    align-items: center;
    border-radius: 20px 20px 0 0;
}
.report-card {
```

```
    background-color: #E0F7FA;
    color: #222;
    border-radius: 8px;
    box-shadow: 0 2px 8px rgba(46, 35, 108, 0.1);
}
.report-file-input {
    margin-bottom: 18px;
    display: block;
}
.report-submit-button {
    background: #2e236c;
    color: #fff;
    border: none;
    border-radius: 10px;
    padding: 10px 28px;
    font-size: 20px;
    cursor: pointer;
}
```

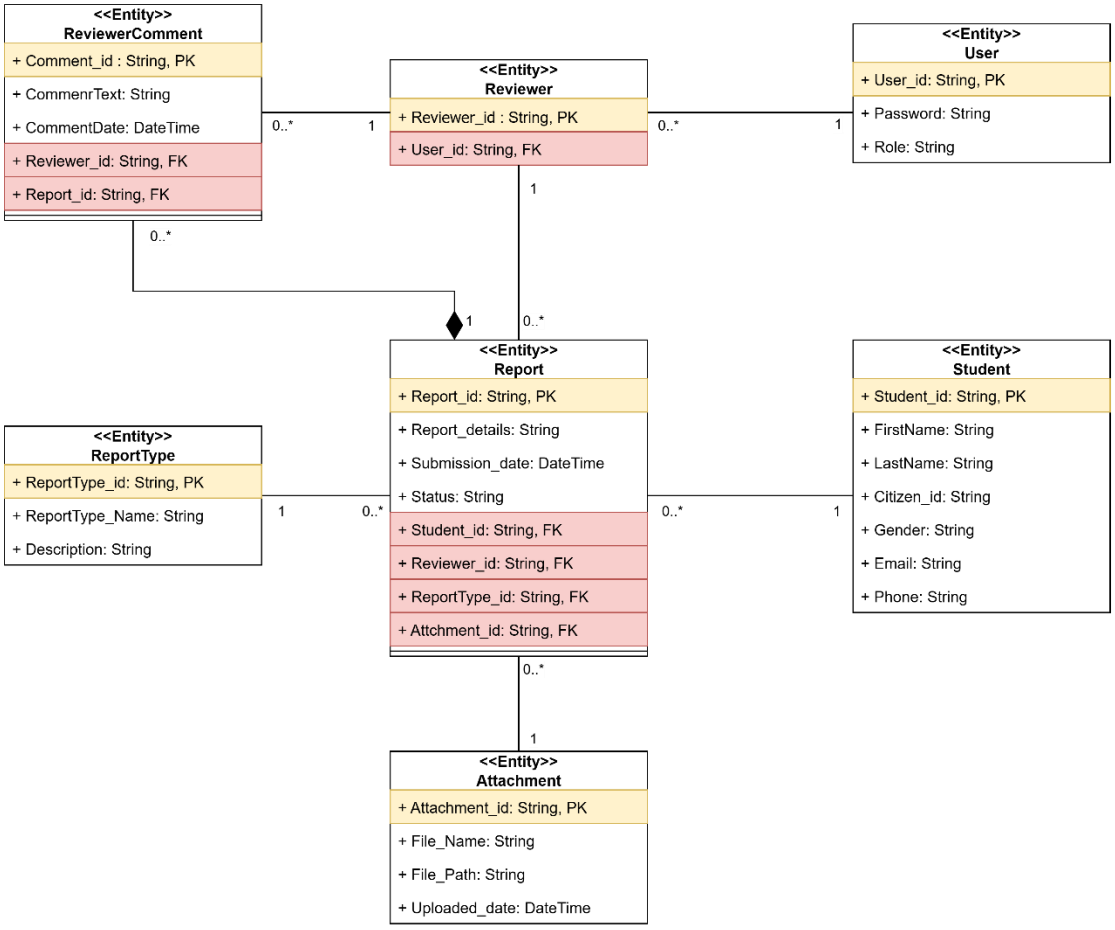


วิเคราะห์ Communication Diagram

Activity	เป็นคำสั่งสำหรับระบบได้หรือไม่	ถ้าเป็น, วัตถุที่รับหน้าที่ทำงานคืออะไร	ชื่อคำสั่ง, ตัวอย่าง parameter
Click “เปิดหน้าจอ”	**เป็นคำสั่ง** สั่งงานเพื่อให้หน้า UI เปิด	:ReportUI	เปิด()
โหลดข้อมูลนักศึกษาที่กำลังใช้งานระบบ	**เป็นคำสั่ง** เกิดการสั่งงานให้โหลดข้อมูลนักศึกษาที่ login อยู่	:ReportController	โหลดข้อมูลนักศึกษา (User_id)
		s:Student	ค้นหาด้วยไอดี (Student_id)
โหลดหน้า ระบบคำร้อง	**เป็นคำสั่ง** สั่งงานเพื่อให้หน้าระบบคำร้องแสดง	:ReportController	เตรียมข้อมูลให้หน้าจอ()
		r:Report	ดึงข้อมูลทั้งหมด()
		rv:Reviewer	
		t:ReportType	
กรอกประเภทคำร้อง	**ไม่เป็นคำสั่ง**	-	-
ระบุคนที่ต้องการส่งคำร้อง	**ไม่เป็นคำสั่ง**	-	-
กรอกรายละเอียดคำร้อง	**ไม่เป็นคำสั่ง**	-	-
กดปุ่ม ยืนยัน	**เป็นคำสั่ง** ระบบทำการจัดเก็บข้อมูลคำร้อง	:ReportController	ยืนยันการเพิ่มคำร้อง(r_details, s_id, rv_id, t_id, Attchment_id)
สร้าง entity Report	**เป็นคำสั่ง**	r:Report	สร้าง(r_id, r_details, Submission_date, Status, s_id, rv_id, t_id, Attachment_id)
บันทึกลง entity Report สำเร็จ	**เป็นคำสั่ง**	r:Report	บันทึก()
แสดงข้อความ "ส่งคำร้องสำเร็จ"	**ไม่เป็นคำสั่ง**	-	-


Communication Diagram ระบบย่อย ระบบคำร้อง
8:ยืนยันการเพิ่มคำร้อง(r_details, s_id, rv_id, t_id, Attachment_id)
10:บันทึก()
4:เตรียมข้อมูลหน้าจอ()
9:สร้าง(r_id, r_details, Submission_date, Status, s_id, rv_id, t_id, Attachment_id)
1:เปิด()
2:โหลดข้อมูลนักศึกษา(User_id)
5:ดึงข้อมูลทั้งหมด
:Student
:ReportUI
:Report Controller
r:Report
6:ดึงข้อมูลทั้งหมด
3:ค้นหาด้วยไอดี(Student_id)
7:ดึงข้อมูลทั้งหมด
rv:Reviewer
s:Student
t:ReportType

Class Diagram at Design Level



ระบบย่อย ระบบลงทะเบียนเรียน

ระบบลงทะเบียนเรียน เป็นระบบย่อยที่พัฒนาขึ้นเพื่อให้นักศึกษาสามารถจัดการเลือกลงทะเบียนเรียนรายวิชาได้ด้วยตนเองอย่างสะดวก รวดเร็ว และมีประสิทธิภาพ โดยเพื่อลดปัญหาการลงทะเบียนผิดพลาด ระบบนี้จึงถูกออกแบบให้มีการรองรับการใช้งานพร้อมกันจำนวนมากในช่วงเปิดเทอม เมื่อนักศึกษาต้องการลงทะเบียนเรียน ระบบจะเปิดให้เลือกวิชาที่เปิดสอนในภาคการศึกษานั้น ได้อย่างอิสระ โดยระบบจะแสดงรายการชื่อวิชา, รหัสวิชา, กลุ่มเรียน, จำนวนหน่วยกิต และช่วงวันเวลาเรียนเวลาสอบ เพื่อให้นักศึกษาตัดสินใจเลือกตามตารางเรียนของตนเอง หลังจากเลือกวิชาและกลุ่มเรียนที่ต้องการแล้ว ระบบจะทำการบันทึกวันและเวลาที่ลงทะเบียน และแสดงผลการลงทะเบียนอย่างชัดเจน เช่น รายการวิชาที่เลือก หน่วยกิตรวม และสถานะการลงทะเบียน ซึ่งช่วยให้นักศึกษาตรวจสอบความถูกต้องก่อนยืนยันอีกครั้ง ระบบยังมีฟังก์ชันในการตรวจสอบเงื่อนไข เช่น ไม่ให้ลงทะเบียนซ้ำรายวิชา, ตรวจสอบหน่วยกิตรวมสูงสุดต่อเทอม, หลีกเลี่ยงเวลาเรียนและสอบซ้อนกัน รวมถึงเชื่อมโยงกับระบบหลักสูตรเพื่อให้แน่ใจว่านักศึกษาเลือกวิชาได้ถูกต้องตามแผนการเรียน นอกจากนี้ ยังมีการเชื่อมโยงกับระบบชำระค่าเล่าเรียน เพื่อคำนวณค่าธรรมเนียมตามจำนวนหน่วยกิตที่ลงทะเบียนไว้ และแสดงสถานะการชำระเงินได้ทันทีหลังการลงทะเบียน

User Story ระบบลงทะเบียนเรียน

ในบทบาทของ (As a) นักศึกษา
ฉันต้องการ (I want to) เลือกรายวิชาและลงทะเบียนเรียน
เพื่อ (So that) สามารถจัดตารางเรียนให้เหมาะสมและตรวจสอบข้อมูลการลงทะเบียนได้อย่างถูกต้อง

Output บนหน้าจอ

- นักศึกษาล็อกอินเข้าสู่ระบบ ⟶ เข้าสู่เมนู “ลงทะเบียนเรียน” ⟶ กรอกรหัสวิชา ⟶ เลือกเพิ่ม หรือลดรายวิชา ⟶ เลือกกลุ่มที่ต้องการเรียน ⟶ กดยืนยัน ⟶ แสดงผลการลงทะเบียน ⟶ ยืนยันการลงทะเบียน ⟶ แสดงตารางสรุปรายวิชที่ลงทะเบียนและวันสอบ

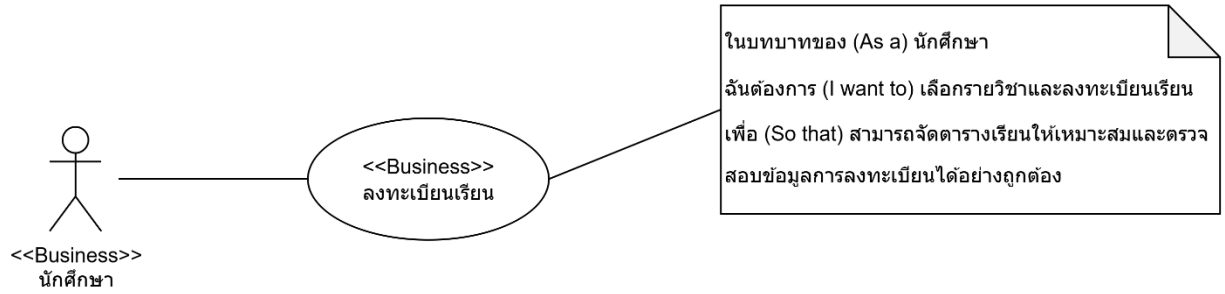
Output ของข้อมูล

ระบบรับข้อมูลการลงทะเบียนเรียน ⟶ ระบบตรวจสอบข้อผิดพลาดก่อนยืนยัน (เช่น หน่วยกิตเกิน, รายวิชาซ้ำ, ตารางชน) ⟶ บันทึกข้อมูลการลงทะเบียนเรียนในฐานข้อมูล

คำนามที่อาจจะกลายมาเป็นตารางในฐานข้อมูลของระบบ

คำนาม	เหตุผล (เกี่ยวข้องกับ Use Case หรือไม่)
นักศึกษา (Student)	เกี่ยวข้องโดยตรง เพราะเป็นผู้ดำเนินการลงทะเบียน
วิชา (Subject)	เกี่ยวข้องโดยตรง เพราะเป็นวิชาที่เปิดให้ลงทะเบียน
รหัสวิชา (Subject Code)	เกี่ยวข้องโดยตรง เพราะใช้ระบุรายวิชา
กลุ่มเรียน (Study Group)	เกี่ยวข้องโดยตรง เพราะใช้ระบุเวลาหรือห้องเรียนของแต่ละวิชา
ภาคการศึกษา (Semester)	เกี่ยวข้องโดยตรง เพราะใช้จำแนกการลงทะเบียนแต่ละภาคการศึกษา
จำนวนหน่วยกิต (Credits)	เกี่ยวข้องโดยตรง เพราะใช้คำนวณค่าเล่าเรียนและภาระการเรียน
วันเวลาที่ลงทะเบียน (Registration Date)	เกี่ยวข้องโดยตรง เพราะใช้ระบุวันที่ลงทะเบียนเพื่อบันทึกและอ้างอิง
การลงทะเบียน (Enrollment)	เกี่ยวข้องโดยตรง เพราะเป็นความสัมพันธ์ระหว่างนักศึกษาและรายวิชาที่เลือก

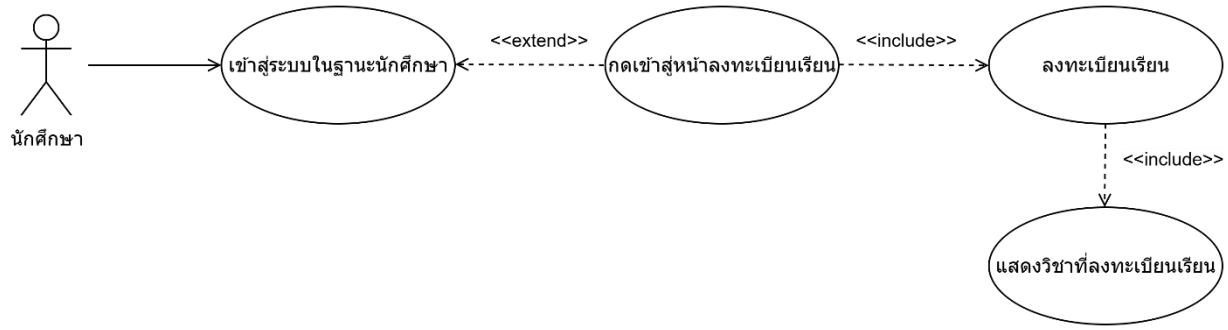
Business Use Case Diagram



Checklist: Business Use Case Diagram

- Business Actor มี<<Business>> กำกับ
- Business Actor เป็นชื่อบทบาท ไม่ใช่ชื่อบุคคล
- Business Use Case มี<<Business>> กำกับ
- Business Use Case ขึ้นต้นด้วยคำที่แสดงพฤติกรรม (คำกริยา)
- พฤติกรรมใน Business Use Case มาจาก User Story
- มี Note ที่แสดง User Story ใน Diagram

System Use Case Diagram



Checklist: System Use Case Diagram

- System Actor และ System Use Case ต้องไม่มีอะไรเชื่อมกัน
- เส้นโยงจาก System Actor ไปยัง System Use Case ต้องมีหัวลูกศรปลายเปิด
- System Actor ต้องเป็นบทบาทของคนที่ใช้งานจริงๆ
- System Use Case ต้องเป็นคำแสดงพฤติกรรม
- หากเป็นการรักษาความปลอดภัย ชื่อ System Use Case จะอยู่ในรูปแบบ "เข้าระบบในฐานะ <Actor>"
- การใช้<<extend>> เป็นเส้นที่หัวลูกศรปลายเปิด ชี้จาก System Use Case ไปยัง Use Case หลัก
- การใช้<<include>> เป็นเส้นที่หัวลูกศรปลายเปิด ชี้จาก System Use Case ไปยัง System Use Case ที่ถูกเชื่อมต่อ

วิเคราะห์ Entity ที่เกี่ยวข้อง

นักศึกษา (Student)

- ชื่อ Entity: Student
- ข้อมูลที่ต้องจัดเก็บ
  - Student_id: เป็น Primary Key เก็บในรูปแบบ varchar ที่ไม่สามารถซ้ำกัน และไม่สามารถเป็นค่า Null ได้
  - FirstName: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - LastName: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Citizen_id: เก็บในรูปแบบ char และไม่สามารถเป็นค่า Null ได้
  - Gender: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Email: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้
  - Phone: เก็บในรูปแบบ varchar และไม่สามารถเป็นค่า Null ได้

STUDENT_ID VERCHAR NOT NULL, PK	**FIRSTNAME** VERCHAR NOT NULL	**LASTNAME** VERCHAR NOT NULL	**CITIZEN_ID** CHAR NOT NULL	**GENDER** VERCHAR NOT NULL	**EMAIL** VARCHAR NOT NULL	**PHONE** VERCHAR NOT NULL
B0000001	สมปอง	สองใจ	1302213456789	ชาย	Sompong@gmail.com	454-564
B0000002	สมหมาย	ดีใจ	1301234567890	ชาย	Sommai@gmail.com	594-412

วิชา(Subject)

- ชื่อ Entity: Subject
- ข้อมูลที่ต้องจัดเก็บ
  - Subject_id: เป็น Primary Key เก็บในรูปแบบของ varchar ที่ไม่สามารถซ้ำกันได้ และไม่สามารถเป็น Null ได้
  - Subject_Name: เก็บในรูปแบบของ varchar และไม่สามารถเป็น Null ได้

- Credits: เก็บในรูปแบบของ integer และไม่สามารถเป็น Null ได้
- Subject_Semester: เก็บในรูปแบบความสัมพันธ์แบบ Many-to-Many โดยมีการสร้างของตารางดังนี้
    - Subject_id: เป็น Primary Key เก็บในรูปแบบของ varchar และต้องไม่เป็นค่า Null
    - Semester_id: เป็น Primary Key เก็บในรูปแบบของ varchar และต้องไม่เป็นค่า Null

SUBJECT_ID VERCHAR NOT NULL, PK	**SUBJECT_NAME** VARCHAR NOT NULL	**CREDITS** INTERGER NOT NULL	**StudyTime** VARCHAR NOT NULL
1	COMPUTER PROGRAMMING I	2	เวลาเรียน: Mo10:00-12:00 B4101 Tu13:00-15:00 B4101 Th13:00-15:00 B2104
2	COMPUTER PROGRAMMING II	2	เวลาเรียน: Tu10:00-12:00 B5203 We17:00-19:00 B1212 Th10:00-12:00 B1211

SUBJECT_ID VERCHAR NOT NULL, PK	**SEMESTER_ID** VERCHAR NOT NULL, PK
1	1
2	1

ภาคการศึกษา (Semester)

- ชื่อ Entity: Semester
- ข้อมูลที่ต้องจัดเก็บ

- Semester_id: เป็น Primary Key เก็บในรูปแบบของ varchar ที่ไม่สามารถซ้ำกันได้และไม่สามารถเป็น Null ได้
- Term: เก็บในรูปแบบของ integer ไม่สามารถเป็น Null ได้
- Academic_Year: เก็บในรูปแบบของ integer และไม่สามารถเป็น Null ได้

SEMESTER _ID VERCHAR NOT NULL, PK	**TERM** INTEGER NOT NULL	**ACADEMIC_YEAR** INTEGER NOT NULL
1	3	2567
2	1	2568

กลุ่มเรียน (Section)

- ชื่อ Entity: กลุ่มเรียน
- ข้อมูลที่ต้องจัดเก็บ
  - Section_id: เป็น Primary Key เก็บในรูปแบบของ varchar ที่ไม่สามารถซ้ำกันได้และไม่สามารถเป็น Null ได้
  - Subject_id: เป็น Foreign Key เก็บในรูปแบบของ varchar ที่ไม่สามารถซ้ำกันได้ และไม่สามารถเป็น Null ได้
  - Group_Name: เก็บในรูปแบบของ varchar และไม่สามารถเป็น Null ได้
  - Schedule: เก็บในรูปแบบของ text และไม่สามารถเป็น Null ได้

SECTION_ID VERCHAR NOT NULL, PK	**SUBJECT_ID** VARCHAR NOT NULL, FK	**GROUP_NAME** VARCHAR NOT NULL	**SCHEDULE** TEXT NOT NULL
12	1	3	เวลาเรียน: Mo10:00-12:00 B4101 Tu13:00-15:00 B4101 Th13:00-15:00 B2104

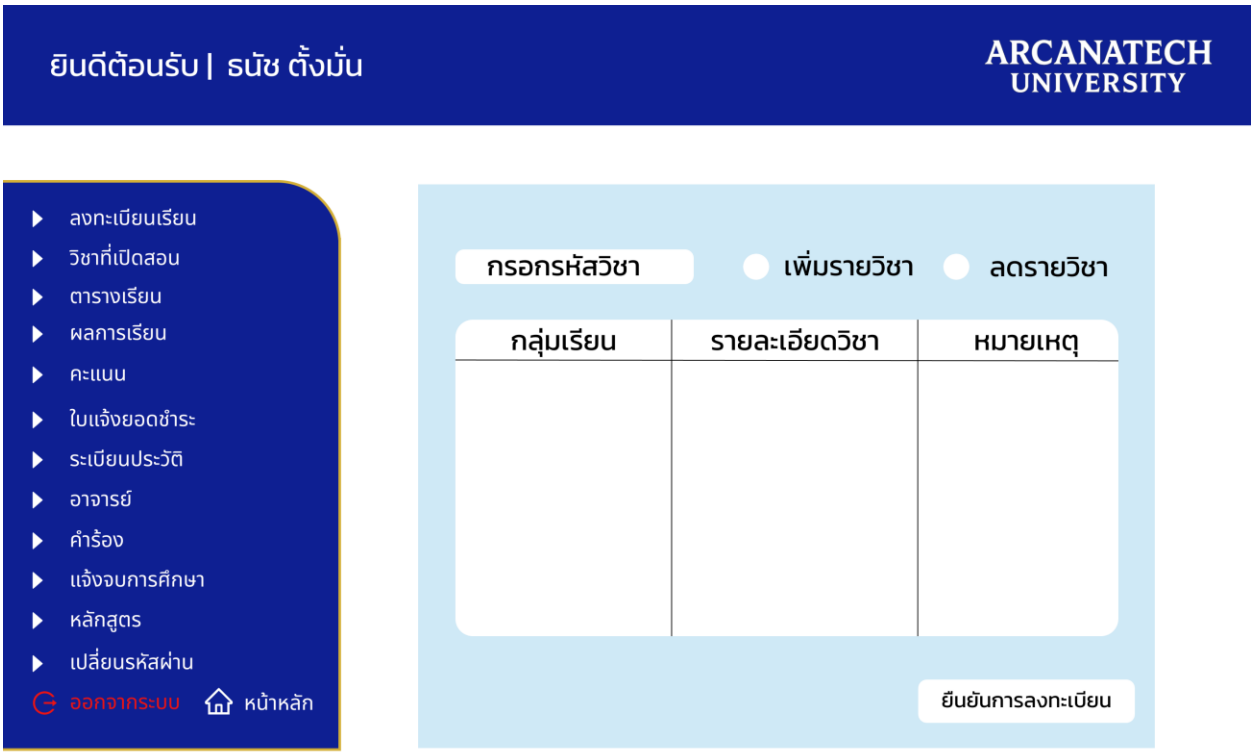
24	2	8	เวลาเรียน: Tu10:00-12:00 B5203 We17:00-19:00 B1212 Th10:00-12:00 B1211

การลงทะเบียน (Registration)

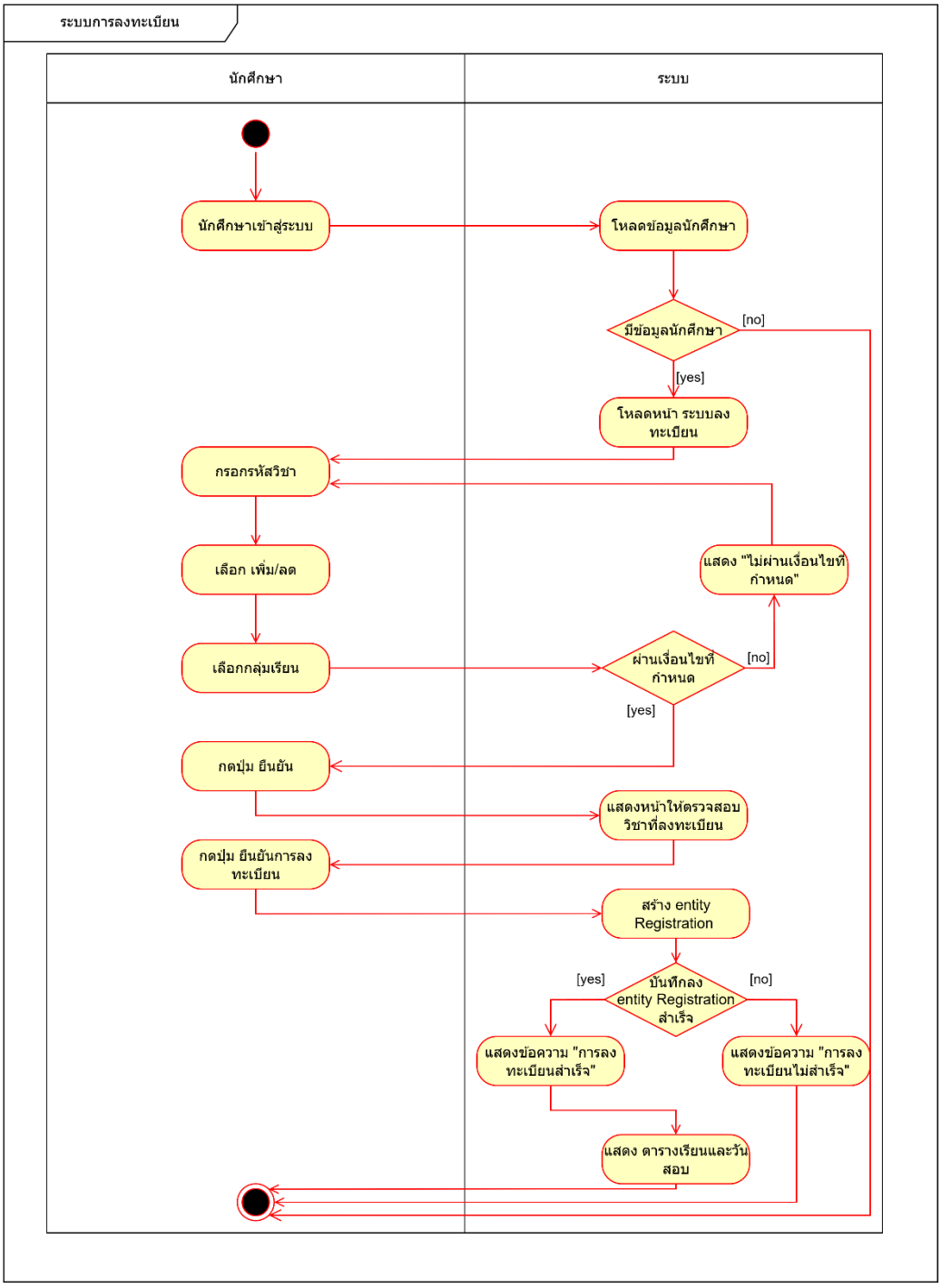
- ชื่อ Entity: Registrations
- ข้อมูลที่ต้องจัดเก็บ
  - Registration_id: เป็น Primary Key เก็บในรูปแบบของ varchar ที่ไม่สามารถซ้ำกันได้ และไม่สามารถเป็น Null ได้
  - Student_id: เป็น Foreign Key เก็บในรูปแบบของ varchar ที่ไม่สามารถซ้ำกันได้ และไม่สามารถเป็น Null ได้
  - Subject_id: เป็น Foreign Key เก็บในรูปแบบของ varchar ที่ไม่สามารถซ้ำกันได้ และไม่สามารถเป็น Null ได้
  - Section_id: เป็น Foreign Key เก็บในรูปแบบของ varchar ที่ไม่สามารถซ้ำกันได้ และไม่สามารถเป็น Null ได้
  - Semester_id: เป็น Foreign Key เก็บในรูปแบบของ integer ที่ไม่สามารถซ้ำกันได้ และไม่สามารถเป็น Null ได้
  - Registration_date: เก็บในรูปแบบของ datetime ที่ไม่สามารถเป็น Null ได้

REGISTRATION_ID VERCHAR NOT NULL, PK	**STUDENT_ID** VERCHAR NOT NULL, FK	**SUBJECT_ID** VERCHAR NOT NULL, FK	**SECTION_ID** VERCHAR NOT NULL, FK	**SEMESTER_ID** VERCHAR NOT NULL, FK	**REGISTRATION_ DATE** DATETIME NOT NULL
1	B0000001	1	12	1	2025-06-29
2	B0000002	2	24	1	2025-06-29

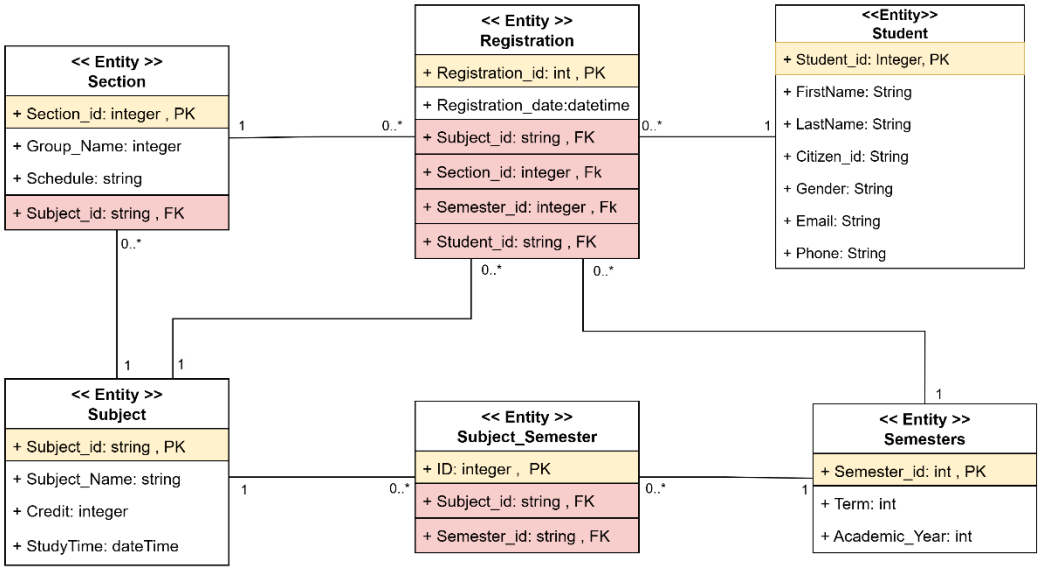
UI Design


ยินดีต้อนรับ | ธนัช ตั้งมั่น
ARCANATECH UNIVERSITY
ลงทะเบียนเรียน
วิชาที่เปิดสอน
ตารางเรียน
ผลการเรียน
คะแนน
ใบแจ้งยอดชำระ
ระเบียนประวัติ
อาจารย์
คำร้อง
แจ้งจบการศึกษา
หลักสูตร
เปลี่ยนรหัสผ่าน
ออกจากระบบ
หน้าหลัก
กรอกรหัสวิชา
เพิ่มรายวิชา
ลดรายวิชา
กลุ่มเรียน
รายละเอียดวิชา
หมายเหตุ
ยืนยันการลงทะเบียน

Activity Diagram



Class Diagram



Checklist for Class Diagram

- ต้องมี Tag <<Entity>> กำกับในส่วนของหัวตาราง
- ในรายละเอียดของ Field ต้องบอกได้ทั้งชื่อ Field และ Datatype ที่จะนำมาใช้งานในการพัฒนาได้จริง Primary Key จะต้องเติม PK ต่อท้าย Datatype
- Foreign Key จะต้องมีกรอบสีแดงกำกับ
- ในส่วนของ Relationship ระหว่าง Entity จะต้องวาดด้วยเส้นตรง และมีตัวเลข 1 และ 0..* ในการกำกับความสัมพันธ์

Backend

Entity – Enrollment.go

```
package entity
import (
    "time"
    "gorm.io/gorm"
)
type Enrollment struct {
    gorm.Model
    Enrollment_id      string     `gorm:"primaryKey" json:"Enrollment_id"`
    Registration_date time.Time `json:"Registration_date"`

    Student_id uint
    Student     Students `gorm:"foreignKey:Student_id"`

    Subject_id uint
    Subject     Subject `gorm:"foreignKey:Subject_id"`

    StudyGroup_id uint
    StudyGroup     StudyGroup `gorm:"foreignKey:StudyGroup_id"`

    Semester_id uint
    Semester     Semester `gorm:"foreignKey:Semester_id"`
}
```

Entity – Subject.go

```
package entity
import (
    "gorm.io/gorm"
)
type Subject struct {
    gorm.Model
    Subject_id    string `gorm:"primaryKey" json:"Subject_id"`
    Subject_Name string `json:"Subject_Name"`
    Credit        int     `json:"Credit"`

    Enrollments []Enrollment `gorm:"foreignKey:Subject_id"`
}
```

Entity – StudyGroup.go

```
package entity
import (
    "gorm.io/gorm"
)
type StudyGroup struct {
    gorm.Model
    StudyGroup_id    string `gorm:"primaryKey" json:"StudyGroup_id"`
    StudyGroup_Name string `json:"StudyGroup_Name"`
    Schedule         string `json:"Schedule"`

    Enrollments []Enrollment `gorm:"StudyGroup_id"`

    Subject_id uint
    Subject    Subject `gorm:"foreignKey:Subject_id"`
}
```

Entity – Semester.go

```
package entity
import (
    "gorm.io/gorm"
)
type Semester struct {
    gorm.Model
    Semester_id    string `gorm:"primaryKey" json:"Semester_id"`
    Term           string `json:"Term"`
    Academic_Year string `json:"Academic_Year"`

    Enrollments []Enrollment `gorm:"foreignKey:Semester_id"`
}
```

FRONTEND

Register.tsx

```
import React, { useState } from 'react';
import { Layout, Input, Radio, Table, Button } from 'antd';
import './register.css';

const { Header, Content, Footer } = Layout;
const wrapperStyle: React.CSSProperties = {
  borderRadius: 8,
  boxShadow: '0 4px 12px rgba(0,0,0,0.08)',
  width: '100%',
  minHeight: '100vh',
  display: 'flex',
  flexDirection: 'column',
  overflow: 'hidden',
};
```

```
const headerStyle: React.CSSProperties = {
  background: '#2e236c',
  color: 'white',
  textAlign: 'center',
  padding: 16,
  fontSize: 20,
  fontWeight: 'bold',
};
const contentStyle: React.CSSProperties = {
  background: '#f5f5f5',
  padding: 24,
  minHeight: 400,
  color: '#333',
  overflowY: 'auto',
};
const footerStyle: React.CSSProperties = {
  background: '#1890ff',
  color: 'white',
  textAlign: 'center',
  padding: 12,
};
const Register: React.FC = () => {
  const [actionType, setActionType] = useState<string>('add');
  const [courseCode, setCourseCode] = useState('');
  const columns = [
    {
      title: 'กลุ่มเรียน',
      dataIndex: 'studygroup',
      key: 'studygroup',
    },
    {
      title: 'รายละเอียดวิชา',
      dataIndex: 'coursedetails',
      key: 'coursedetails',
    },
```

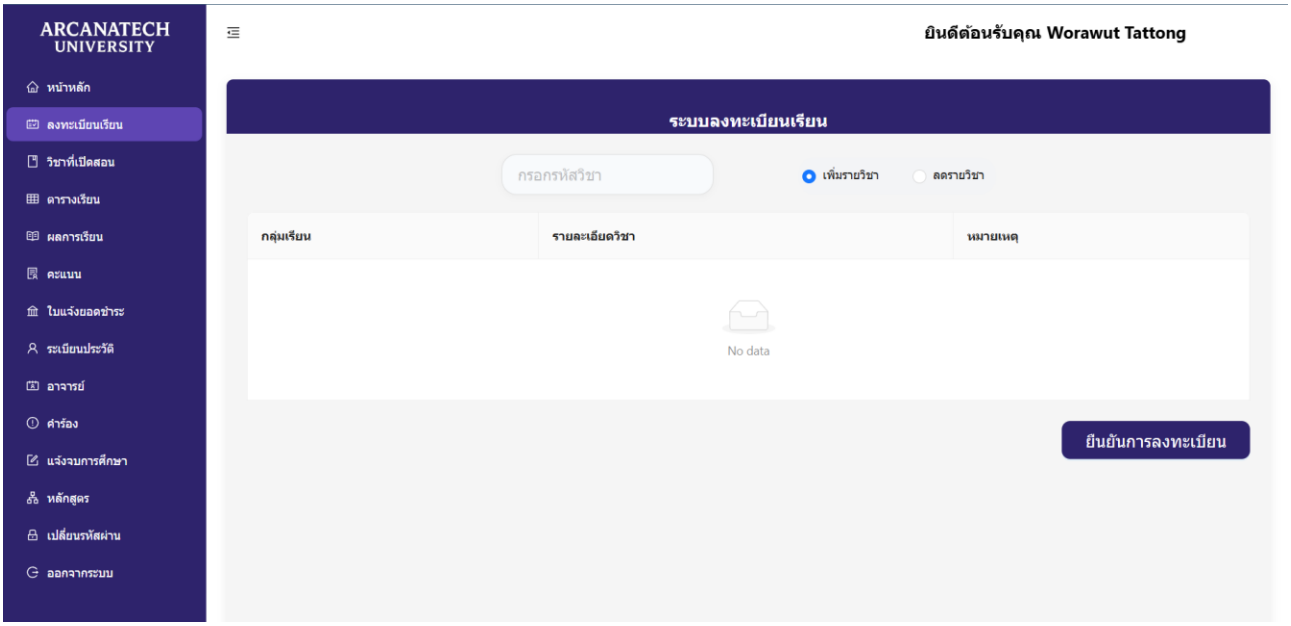
```
    {
      title: 'หมายเหตุ',
      dataIndex: 'note',
      key: 'note',
    },
  ];
  const data: any[] = [];
  return (
    <Layout style={wrapperStyle}>
      <Header style={headerStyle}>ระบบลงทะเบียนเรียน</Header>
      <Content style={contentStyle}>
        <div style={{ display: 'flex', gap: 100, flexWrap: 'wrap', justifyContent: 'center', marginBottom: 20
}}>
          <Input
            className="register-input"
            placeholder="กรอกรหัสวิชา"
            style={{ width: 250 }}
            value={courseCode}
            onChange={(e) => setCourseCode(e.target.value)}
          />
          <Radio.Group
            className="add-drop-radio"
            value={actionType}
            onChange={(e) => setActionType(e.target.value)}
            style={{ marginTop: 4 }}
          >
            <Radio value="add">เพิ่มรายวิชา</Radio>
            <Radio value="drop">ลดรายวิชา</Radio>
          </Radio.Group>
        </div>
        <Table
          columns={columns}
          dataSource={data}
          pagination={false}
          bordered
```

```
          style={{ backgroundColor: "white", borderRadius: 8 }}
        />
        <div style={{ display: 'flex', justifyContent: 'flex-end', marginTop: 24 }}>
          <Button type="primary" className="register-submit-button">
        ยืนยันการลงทะเบียน
          </Button>
        </div>
      </Content>
      <Footer style={footerStyle}>Footer © 2025</Footer>
    </Layout>
  );
};
export default Register;
```

Register.css

```
.register-input {
  width: 230px;
  height: 48px;
  border-radius: 18px;
  font-size: 18px;
  border: 1.5px solid #e0e0e0;
  padding: 0 18px;
  background: #f8fafc;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.03);
  transition: border 0.2s;
}
.add-drop-radio {
  display: flex;
  gap: 24px;
  background: #f3f4f6;
  border-radius: 20px;
  padding: 9px 5px;
}
.register-submit-button {
```

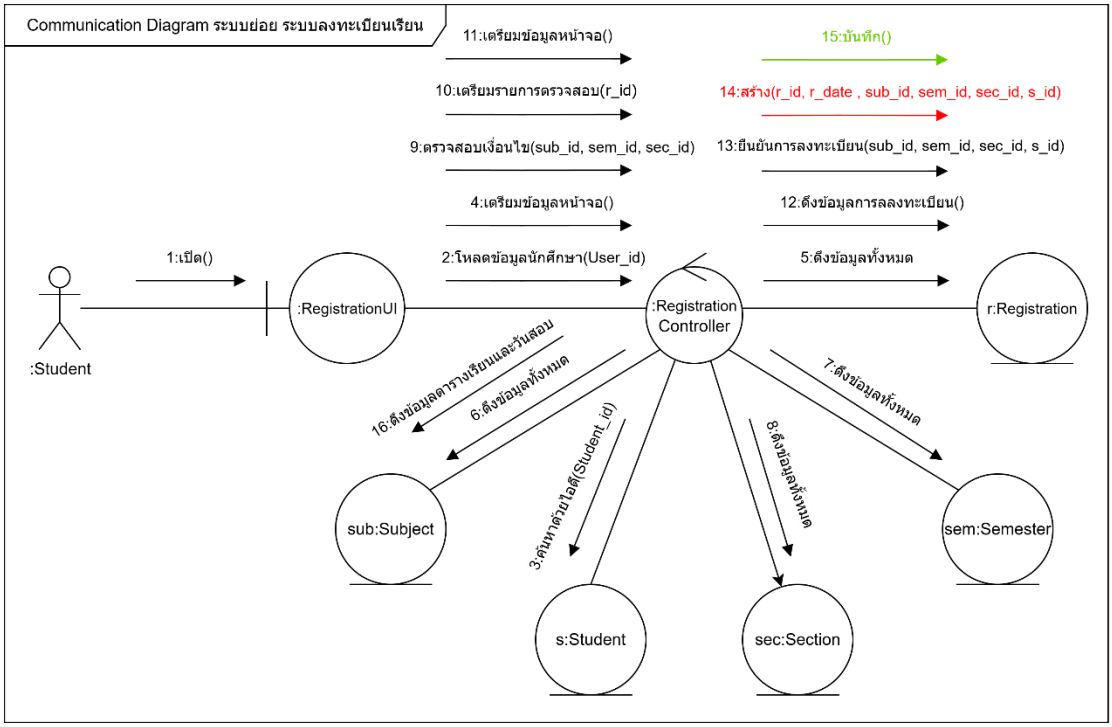
```
  height: 45px;
  background: #2e236c;
  color: #fff;
  border: none;
  border-radius: 10px;
  padding: 10px 28px;
  font-size: 20px;
  cursor: pointer;
}
```



วิเคราะห์ Communication Diagram

Activity	เป็นคำสั่งสำหรับระบบได้หรือไม่	ถ้าเป็น, วัตถุที่รับหน้าที่ทำงานคืออะไร	ชื่อคำสั่ง, ตัวอย่าง parameter
Click “เปิดหน้าจอ”	**เป็นคำสั่ง** สั่งงานเพื่อให้หน้า UI เปิด	:RegistrationUI	เปิด()
โหลดข้อมูลนักศึกษาที่กำลังใช้งานระบบ	**เป็นคำสั่ง** เกิดการสั่งงานให้โหลดข้อมูลนักศึกษาที่ login อยู่	:RegistrationController	โหลดข้อมูลนักศึกษา (User_id)
		s:Student	ค้นหาด้วยไอดี (Student_id)
โหลดหน้า ระบบลงทะเบียนเรียน	**เป็นคำสั่ง** สั่งงานเพื่อให้หน้าระบบลงทะเบียนเรียน	:RegistrtionController	เตรียมข้อมูลให้หน้าจอ()
		r:Registration	ดึงข้อมูลทั้งหมด()
		sub:Subject	
		sem:Semester	
		sec:Section	
กรอกรหัสวิชา	**ไม่เป็นคำสั่ง**	-	-
เลือก เพิ่ม/ลด	**ไม่เป็นคำสั่ง**	-	-
เลือกกลุ่มเรียน	**ไม่เป็นคำสั่ง**	-	-
ผ่านเงื่อนไขที่กำหนด	**เป็นคำสั่ง** ตรวจสอบเงื่อนไขการลงทะเบียน	:RegistrtionController	ตรวจสอบเงื่อนไข(sub_id, sem_id, sec_id)
กดปุ่ม ยืนยัน	**เป็นคำสั่ง**	:RegistrationController	เตรียมรายการตรวจสอบ(r_id)
แสดงหน้าให้ตรวจสอบวิชาที่ลงทะเบียน	**เป็นคำสั่ง**	:RegistrtionController	เตรียมข้อมูลให้หน้าจอ()
		r:Registration	ดึงข้อมูลการลลงทะเบียน()
กดปุ่ม ยืนยันการลงทะเบียน	**เป็นคำสั่ง** ระบบทำการจัดเก็บข้อมูลการลงทะเบียน	r:Registration	ยืนยันการลงทะเบียน (sub_id, sem_id, sec_id, s_id)

สร้าง entity Registration	**เป็นคำสั่ง**	r:Registration	สร้าง(r_id, r_date , sub_id, sem_id, sec_id, s_id)
บันทึกลง entity Registrationสำเร็จ	**เป็นคำสั่ง**	r:Registration	บันทึก()
แสดงข้อความ "การลงทะเบียนสำเร็จ"	**ไม่เป็นคำสั่ง**	-	-
แสดง ตารางเรียนและวันสอบ	**เป็นคำสั่ง**	sub:subject	ดึงข้อมูลตารางเรียนและวันสอบ



Class Diagram at Design Level