A red and white circular logo

Description automatically generated

**โครงงาน**

Mini Project

**จัดทำโดย**

6504062610137 นายธีราพัฒน์ จิรภาสนิธิเลิศ

**เสนอ**

ผู้ช่วยศาสตราจารย์สถิต ประสมพันธ์

วิชา Object Oriented Programming

ภาคเรียนที่ 1/2565

ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ประยุกต์

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

**เกี่ยวกับโครงงาน**

ชื่อโปรเจค: Numerical Project

นำเสนอโดย: นายธีราพัฒน์ จิรภาสนิธิเลิศ

อาจารย์ผู้สอน: ผู้ช่วยศาสตราจารย์สถิต ประสมพันธ์

**บทที่ 1 ที่มาและความสําคัญของโครงการ**

โครงงานนี้จัดขึ้นเพื่อวัดผลความสามารถในการเรียนวิชา Object Oriented Programming โดย

การนําเรื่องที่เรียนมาสร้างเป็นชิ้นงานในรูปแบบ website เพิ่อเข้าใจกหลักการทำงาน algoristm ต่างๆในวิชา

Numerical method และ การเขียนโปรแกรม Object Oriented Programming เพื่อให้ง่ายต่อการออกแบบโครงสร้าง website

**ประเภทโครงการ**

Web Development

**ประโยชน์**

1. เป็นแหล่งเผยแพร่ อัลกอลิทึมวิชา numerical method

2. Web Application ช่วยคำนวณหาค่า รากสมการ

3. Web Application ฝึกทำโจทย์ หารากสมการ จากการกดสุ่มสการพร้อมคำตอบ

**ขอบเขตโครงการ**

ตารางแผนการทำงานเดือนตุลาคม-พฤศจิกายน

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ลำดับ | รายการ | 1-5 | 6-7 | 8-25 | 26-13 |
| 1 | วางแผน และ ออกแบบ |  |  |  |  |
| 2 | ศึกษาข้อมูลต่างๆ |  |  |  |  |
| 3 | เขียนโปรแกรม |  |  |  |  |
| 4 | จัดทำเอกสาร |  |  |  |  |
| 5 | ตรวจสอบข้อผิดพลาด |  |  |  |  |

**บทที่ 2 ส่วนการพัฒนา**

**รูปแบบการพัฒนา**

**Design**

1. ศึกษาอัลกอลิทึมทั้งหมดในวิชา numerical method
2. ออกแบบโครงสร้าง และ ตาราง Database
3. ออกแบบโครงสร้างโปรแกรม
4. กำหนด เครื่องมือ Ide ที่ใช้ และ ภาษาที่ใช้

**Implement**

1. ลงมือทำตามแผนทีวางไว้
2. Testing website ทุก page หลัง page เสร็จทีละ page

**Testing**

ทดสอบ ทุก webpage และ การเชื่อมต่อ Database

Maintenance

ปรับปรุงแก้ไข ข้อผิดพลาดการทำงาน ของ Web Application

โครงสร้าง Web Application

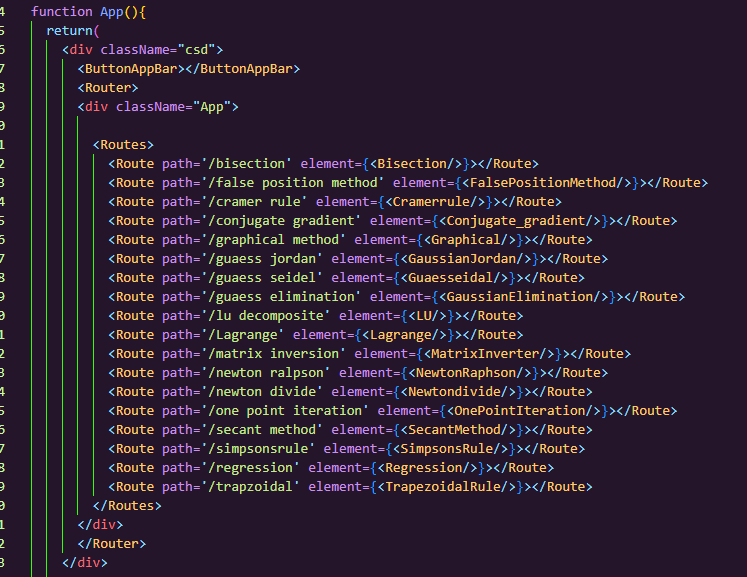
Component หลัก ชื่อ App ไฟล์ main.jsx

A screenshot of a computer program

Description automatically generated

ตรวจจับข้อผิดพลาดด้วย React.StrictMode

Component App



สร้าง Composition React Routes เพื่อสร้างเป็น Multi Application เชื่อมต่อ ทุกๆ Single Application หรือ แต่ละหน้า Web Application โดย มี React Route ใช้ในการกำหนด UI แต่ละชิ้นที่ควรจะแสดงเมื่อ URL ตรงกับเส้นทางที่กำหนด

**โปรแกรม Object-Oriented**

**Class Bisection**

**Constructor**

class Bisection extends React.Component {

    maxiteration = 1000000;

    database\_equation = collection(db, "find\_rootequation");

    constructor(props) {

        super(props);

        this.state = {

            datafunction: [],

            datamin: [],

            datamax: [],

            a: 0,

            b: 0,

            ans: 'pow(7,4)-x',

            tolerance: 0.0001,

            result: null,

            val: 'pow(7,4)-x',

            x: 0,

            y: 0

        };

    }

**Inheritance**

Class Bisection สืบทอด จาก React.Component เพื่อดึงคุณสมบัติ Component มาใช้

class Bisection extends React.Component {

**Composition**

 <Show\_graph func={val} x={x} y={y} />

**Encapsulation**

React, state และ lifecycle methods เป็นตัวอย่างของ Encapsulation. State ไม่ควรเปลี่ยนแปลงโดยตรงนอกเหนือจากเมธอดที่ React จัดหาไว้

this.state = {

            datafunction: [],

            datamin: [],

            datamax: [],

            a: 0,

            b: 0,

            ans: 'pow(7,4)-x',

            tolerance: 0.0001,

            result: null,

            val: 'pow(7,4)-x',

            x: 0,

            y: 0

        };

**Class Conjugategradient**

**Constructor**

 constructor(){

        super();

        this.state = {

            dimension:2,

            matrix: Array.from({ length: 2 }, () => Array(3).fill(0)),

            answer: [],

            numIterations: 0,

            database\_equation: collection(db,"Conjugate\_gradient"),

            all\_dimension:[],

            y:[],

            x:[]

        }

        this.setmatrix = this.setmatrix.bind(this);

        this.random\_matrix = this.random\_matrix.bind(this);

        this.check\_duplicate = this.check\_duplicate.bind(this);

        this.submitdata = this.submitdata.bind(this);

        this.handleDimensionChange = this.handleDimensionChange.bind(this);

        this.handleMatrixChange = this.handleMatrixChange.bind(this);

        this.dotProduct = this.dotProduct.bind(this);

        this.matrixVectorMult = this.matrixVectorMult.bind(this);

        this.subtractVectors = this.subtractVectors.bind(this);

        this.addVectors = this.addVectors.bind(this);

        this.scalarMult = this.scalarMult.bind(this);

        this.calculate = this.calculate.bind(this);

    }

**Inheritance**

Class Conjugate\_gradient สืบทอด จาก React.Component เพื่อดึงคุณสมบัติ Component มาใช้

export default class Conjugate\_gradient extends React.Component{

**Encapsulation**

React, state และ lifecycle methods เป็นตัวอย่างของ Encapsulation. State ไม่ควรเปลี่ยนแปลงโดยตรงนอกเหนือจากเมธอดที่ React จัดหาไว้

 this.state = {

            dimension:2,

            matrix: Array.from({ length: 2 }, () => Array(3).fill(0)),

            answer: [],

            numIterations: 0,

            database\_equation: collection(db,"Conjugate\_gradient"),

            all\_dimension:[],

            y:[],

            x:[]

        }

**Class Cramerrule**

**Constructor**

constructor() {

    super();

    this.state = {

      dimension: 2,

      matrix: Array.from({ length: 2 }, () => Array(3).fill(0)),

      answer: []

    };

  }

**Inheritance**

Class Cramerrule สืบทอด จาก React.Component เพื่อดึงคุณสมบัติ Component มาใช้

class Cramerrule extends Component

**Encapsulation**

React, state และ lifecycle methods เป็นตัวอย่างของ Encapsulation. State ไม่ควรเปลี่ยนแปลงโดยตรงนอกเหนือจากเมธอดที่ React จัดหาไว้

 this.state = {

      dimension: 2,

      matrix: Array.from({ length: 2 }, () => Array(3).fill(0)),

      answer: []

    };

**Class FalsePostionMethod**

**Constructor**

 constructor() {

    super();

    this.state = {

      a: 0, // Initial lower bound

      b: 1, // Initial upper bound

      ans: "",

      root: 1,

      tolerance: 0.00001, // Tolerance for stopping criterion

      maxIterations: 100, // Maximum number of iterations

      result: null, // The approximate root

    };

  }

**Inheritance**

Class FalsePostionMethod สืบทอด จาก React.Component เพื่อดึงคุณสมบัติ Component มาใช้

class FalsePositionMethod extends Component {

**Encapsulation**

React, state และ lifecycle methods เป็นตัวอย่างของ Encapsulation. State ไม่ควรเปลี่ยนแปลงโดยตรงนอกเหนือจากเมธอดที่ React จัดหาไว้

this.state = {

      a: 0, // Initial lower bound

      b: 1, // Initial upper bound

      ans: "",

      root: 1,

      tolerance: 0.00001, // Tolerance for stopping criterion

      maxIterations: 100, // Maximum number of iterations

      result: null, // The approximate root

    };

**Class Graphical**

**Constructor**

constructor(props) {

    super(props);

    this.state = {

      answer: 1,

      equation: "",

      datafunction: []

    };

    this.database\_equation = collection(db, "Graphical");

  }

**Inheritance**

Class Graphical สืบทอด จาก React.Component เพื่อดึงคุณสมบัติ Component มาใช้

class Graphical extends React.Component {

**Encapsulation**

React, state และ lifecycle methods เป็นตัวอย่างของ Encapsulation. State ไม่ควรเปลี่ยนแปลงโดยตรงนอกเหนือจากเมธอดที่ React จัดหาไว้

    this.state = {

      answer: 1,

      equation: "",

      datafunction: []

    };

**Composition**

สร้าง Component Graph เพื่อแสดง Graph

 <Mafs>

          <Coordinates.Cartesian />

          <Plot.OfX y={(x) => eval(equation)} color={Theme.blue} />

        </Mafs>

**Class GaussuanElimination**

**Constructor**

 constructor(){

        super();

        this.state = {

            desiredRowCount: 3,

            matrix: [],

            solution: [],

            errorMessage: '',

            database\_equation: collection(db,"guessian"),

            all\_dimension: [],

            y:[],

            x:[]

        }

**Inheritance**

Class GaussianElimination สืบทอด จาก React.Component เพื่อดึงคุณสมบัติ Component มาใช้

export default class GaussianElimination extends React.Component{

**Encapsulation**

React, state และ lifecycle methods เป็นตัวอย่างของ Encapsulation. State ไม่ควรเปลี่ยนแปลงโดยตรงนอกเหนือจากเมธอดที่ React จัดหาไว้

 constructor(){

        super();

        this.state = {

            desiredRowCount: 3,

            matrix: [],

            solution: [],

            errorMessage: '',

            database\_equation: collection(db,"guessian"),

            all\_dimension: [],

            y:[],

            x:[]

        }

**Class Guaessjordan**

**Constructor**

 constructor(){

        super();

        this.state = {

            desiredRowCount:3,

            matrix:Array(3).fill().map(() => Array(4).fill(0)),

            solution:[],

            errorMessage:'',

            database\_equation:collection(db,"guessian"),

            y:[],

            x:[],

            all\_dimension:[]

        }

        this.setmatrix = this.setmatrix.bind(this);

        this.random\_matrix = this.random\_matrix.bind(this);

        this.submitdata = this.submitdata.bind(this);

        this.solve = this.solve.bind(this);

        this.handleInputChange = this.handleInputChange.bind(this);

        this.handleSizeChange = this.handleSizeChange.bind(this);

        this.initializeMatrix = this.initializeMatrix.bind(this);

    }

**Inheritance**

Class Guaessjordan สืบทอด จาก React.Component เพื่อดึงคุณสมบัติ Component มาใช้

export default class GaussianJordan extends React.Component{

**Encapsulation**

React, state และ lifecycle methods เป็นตัวอย่างของ Encapsulation. State ไม่ควรเปลี่ยนแปลงโดยตรงนอกเหนือจากเมธอดที่ React จัดหาไว้

this.state = {

            desiredRowCount:3,

            matrix:Array(3).fill().map(() => Array(4).fill(0)),

            solution:[],

            errorMessage:'',

            database\_equation:collection(db,"guessian"),

            y:[],

            x:[],

            all\_dimension:[]

        }

**Class Guaesseidal**

**Constructor**

constructor(props) {

        super(props);

        this.state = {

            dimension: 2,

            matrix: Array.from({ length: 2 }, () => Array(3).fill(0)),

            answer: [],

            all\_dimension: [],

            y: [],

            x: []

        };

        this.database\_equation = collection(db, "guessian");

    }

**Inheritance**

Class Guaesseidal สืบทอด จาก React.Component เพื่อดึงคุณสมบัติ Component มาใช้

class Guaesseidal extends React.Component {

**Encapsulation**

React, state และ lifecycle methods เป็นตัวอย่างของ Encapsulation. State ไม่ควรเปลี่ยนแปลงโดยตรงนอกเหนือจากเมธอดที่ React จัดหาไว้

this.state = {

            dimension: 2,

            matrix: Array.from({ length: 2 }, () => Array(3).fill(0)),

            answer: [],

            all\_dimension: [],

            y: [],

            x: []

        };

**Class Lagrange**

**Constructor**

constructor(props) {

        super(props);

        this.state = {

            matrixSize: 2,

            matrixData: Array(2).fill(Array(2).fill('')),

            outputData: [],

            pointsize: 1,

            pointdata: [],

            Fx: 0,

            Y: undefined

        };

    }

**Inheritance**

Class Lagrange สืบทอด จาก React.Component เพื่อดึงคุณสมบัติ Component มาใช้

class Lagrange extends Component {

**Encapsulation**

React, state และ lifecycle methods เป็นตัวอย่างของ Encapsulation. State ไม่ควรเปลี่ยนแปลงโดยตรงนอกเหนือจากเมธอดที่ React จัดหาไว้

this.state = {

            matrixSize: 2,

            matrixData: Array(2).fill(Array(2).fill('')),

            outputData: [],

            pointsize: 1,

            pointdata: [],

            Fx: 0,

            Y: undefined

        };

**Class Lu**

**Constructor**

constructor(props) {

        super(props);

        this.state = {

            matrixSize: 2,

            matrixData: Array(2).fill(Array(2 + 1).fill('')),

            outputData: [],

            X: []

        };

    }

**Inheritance**

Class Lu สืบทอด จาก React.Component เพื่อดึงคุณสมบัติ Component มาใช้

class LU extends Component {

**Encapsulation**

React, state และ lifecycle methods เป็นตัวอย่างของ Encapsulation. State ไม่ควรเปลี่ยนแปลงโดยตรงนอกเหนือจากเมธอดที่ React จัดหาไว้

this.state = {

            matrixSize: 2,

            matrixData: Array(2).fill(Array(2 + 1).fill('')),

            outputData: [],

            X: []

        };

**Class MatrixcInverter**

**Constructor**

constructor(props) {

        super(props);

        this.state = {

            size: 2,

            matrix: [[1, 0], [0, 1]],

            inverseMatrix: null,

            all\_dimension: [],

            x: [],

        };

        this.database\_equation = collection(db, "matrix\_inversion");

    }

**Inheritance**

Class MatrixcInverter สืบทอด จาก React.Component เพื่อดึงคุณสมบัติ Component มาใช้

class MatrixInverter extends React.Component {

**Encapsulation**

React, state และ lifecycle methods เป็นตัวอย่างของ Encapsulation. State ไม่ควรเปลี่ยนแปลงโดยตรงนอกเหนือจากเมธอดที่ React จัดหาไว้

 this.state = {

            size: 2,

            matrix: [[1, 0], [0, 1]],

            inverseMatrix: null,

            all\_dimension: [],

            x: [],

        };

**Class NewtonRaphson**

**Constructor**

constructor(){

        super();

        this.state = {

            max\_iteration: 10000,

            initialGuess:0,

            ans:'',

            func:'',

            tolerance:0.0001,

            result:null,

            datafunction:[],

            guess:[],

            x:null,

            y:null,

            database\_equation:collection(db,"newton\_ralpson")

        };

**Inheritance**

Class NewtonRalpson สืบทอด จาก React.Component เพื่อดึงคุณสมบัติ Component มาใช้

class NewtonRaphson extends React.Component{

**Encapsulation**

React, state และ lifecycle methods เป็นตัวอย่างของ Encapsulation. State ไม่ควรเปลี่ยนแปลงโดยตรงนอกเหนือจากเมธอดที่ React จัดหาไว้

this.state = {

            max\_iteration: 10000,

            initialGuess:0,

            ans:'',

            func:'',

            tolerance:0.0001,

            result:null,

            datafunction:[],

            guess:[],

            x:null,

            y:null,

            database\_equation:collection(db,"newton\_ralpson")

        };

**Class Newtondivide**

**Constructor**

constructor(){

        super();

        this.state = {

            numberdata: 2,

            numberpoint: 2,

            matrixX: [],

            matrixY: [],

            point: [0,0],

            answer: 0,

            selectOption: "Linear Interpolation"

        };

        this.handleDimensionChange\_matrix = this.handleDimensionChange\_matrix.bind(this);

        this.handleOptionChange = this.handleOptionChange.bind(this);

        this.handleOptionChange\_x = this.handleOptionChange\_x.bind(this);

        this.handleInputChanges\_point = this.handleInputChanges\_point.bind(this);

        this.handleInputChanges\_matrix\_x = this.handleInputChanges\_matrix\_x.bind(this);

        this.handleInputChanges\_matrix\_y = this.handleInputChanges\_matrix\_y.bind(this);

        this.setMatrixs = this.setMatrixs.bind(this);

        this.setpoint = this.setpoint.bind(this);

        this.select = this.select.bind(this);

        this.linear\_interpolation = this.linear\_interpolation.bind(this);

        this.quadratic\_interpolation = this.quadratic\_interpolation.bind(this);

        this.polynomial\_interpolation = this.polynomial\_interpolation.bind(this);

        this.interpolationed = this.interpolationed.bind(this);

    }

**Inheritance**

Class newtondivide สืบทอด จาก React.Component เพื่อดึงคุณสมบัติ Component มาใช้

export default class Newtondivide extends React.Component{

**Encapsulation**

React, state และ lifecycle methods เป็นตัวอย่างของ Encapsulation. State ไม่ควรเปลี่ยนแปลงโดยตรงนอกเหนือจากเมธอดที่ React จัดหาไว้

 this.setState({

            numberdata: newDimension,

            matrixX: Array.from({ length: newDimension }, () => Array(newDimension).fill(0)),

            matrixY: Array.from({ length: newDimensions }, () => Array(newDimensions).fill(0))

        });

**Composition**

สร้าง Component Polynomial\_point\_ui และ Component Linearquadratric\_point\_ui เพื่อแสดง Gui polynomial newton divide function และ Linear newton divide function

    <Polynomial\_point\_ui

               selectOption={selectOption}

               numberdata={numberdata}

               handleOptionChange\_x={this.handleOptionChange\_x}

              />

              <Linearquadratic\_point\_ui

               point={this.state.point}

               numberdata={numberdata}

               handleInputChanges\_point={this.handleInputChanges\_point}

               select={this.select}

              />

**Class OnePointIteration**

**Constructor**

 constructor() {

    super();

    this.state = {

      initialGuess: 0,

      tolerance: 0.0001,

      ans: "1/43-x",

      func: "1/43-x",

      result: null,

      xe: 0,

      allfunc: [],

      guess: []

    };

  }

**Inheritance**

Class OnePointIteration สืบทอด จาก React.Component เพื่อดึงคุณสมบัติ Component มาใช้

class OnePointIteration extends Component {

**Encapsulation**

React, state และ lifecycle methods เป็นตัวอย่างของ Encapsulation. State ไม่ควรเปลี่ยนแปลงโดยตรงนอกเหนือจากเมธอดที่ React จัดหาไว้

this.state = {

      initialGuess: 0,

      tolerance: 0.0001,

      ans: "1/43-x",

      func: "1/43-x",

      result: null,

      xe: 0,

      allfunc: [],

      guess: []

    };

**Composition**

สร้าง Component Graph เพื่อแสดง Graph

<Show\_graph func={this.state.func} x={this.state.xe} y={this.g(this.state.xe)} />

**Class Regression**

**Constructor**

constructor(){

      super();

        this.state={

            numberdata:1,

            matrix:[],

            matrix\_y:[],

            selectedOption:" ",

            answer:[],

            findx:0,

            m:0,

            x:0,

            matrix\_x:Array.from({ length: 0 }, () => Array(numberdata).fill(0))

        }

        this.handleOptionChange = this.handleOptionChange.bind(this);

        this.handleInputChanges\_matrix = this.handleInputChanges\_matrix.bind(this);

        this.handleInputChanges\_matrix\_x = this.handleInputChanges\_matrix\_x.bind(this);

        this.handleInputChanges\_matrix\_y = this.handleInputChanges\_matrix\_y.bind(this);

        this.setMatrixs = this.setMatrixs.bind(this);

        this.select = this.select.bind(this);

        this.linear\_regression = this.linear\_regression.bind(this);

        this.polynomial\_regression = this.polynomial\_regression.bind(this);

        this.multi\_linear\_regression = this.multi\_linear\_regression.bind(this);

        this.setmatrix = this.setmatrix;

        this.addx = this.addx.bind(this);

        this.addxy = this.addxy.bind(this);

        this.determinant = this.determinant.bind(this);

        this.setx = this.setx.bind(this);

        this.setm = this.setm.bind(this);

    };

**Inheritance**

Class Regression สืบทอด จาก React.Component เพื่อดึงคุณสมบัติ Component มาใช้

 class Regression extends React.Component{

**Encapsulation**

React, state และ lifecycle methods เป็นตัวอย่างของ Encapsulation. State ไม่ควรเปลี่ยนแปลงโดยตรงนอกเหนือจากเมธอดที่ React จัดหาไว้

 this.state={

            numberdata:1,

            matrix:[],

            matrix\_y:[],

            selectedOption:" ",

            answer:[],

            findx:0,

            m:0,

            x:0,

            matrix\_x:Array.from({ length: 0 }, () => Array(numberdata).fill(0))

        }

**Composition**

**สร้าง Component Inputconfirm เพื่อ สร้าง Gui กำหนดการทำงานโปรแกรม หาก ต้องการคำนวณ Multi regression หรือ Polynomial regression**

 <Inputconfirm

                    m ={this.state.m}

                    x={this.state.x}

                    setx={this.setx}

                    setm={this.setm}

                    selectOption={this.state.selectedOption}

                    matrix\_x={this.state.matrix\_x}

                    matrix\_y={this.state.matrix\_y}

                    matrix={this.state.matrix}

                    select={this.select}

                    handleInputChanges\_matrix={this.handleInputChanges\_matrix}

                    handleInputChanges\_matrix\_x={this.handleInputChanges\_matrix\_x}

                    handleInputChanges\_matrix\_y={this.handleInputChanges\_matrix\_y}/>

                </div>

**Class SecantMethod**

**Constructor**

constructor() {

    super();

    this.state = {

      x0: 1, // Initial guess x0

      x1: 2, // Initial guess x1

      ans: " ",

      tolerance: 0.00001, // Tolerance for stopping criterion

      maxIterations: 100, // Maximum number of iterations

      result: null, // The approximate root

    };

  }

**Inheritance**

Class SecantMethod สืบทอด จาก React.Component เพื่อดึงคุณสมบัติ Component มาใช้

class SecantMethod extends Component {

**Encapsulation**

React, state และ lifecycle methods เป็นตัวอย่างของ Encapsulation. State ไม่ควรเปลี่ยนแปลงโดยตรงนอกเหนือจากเมธอดที่ React จัดหาไว้

this.state = {

      x0: 1, // Initial guess x0

      x1: 2, // Initial guess x1

      ans: " ",

      tolerance: 0.00001, // Tolerance for stopping criterion

      maxIterations: 100, // Maximum number of iterations

      result: null, // The approximate root

    };

**Class SimsonsRule**

**Constructor**

constructor() {

    super();

    this.state = {

      funcString: 'x\*x',

      func: 'x\*x',

      lowerLimit: 0,

      upperLimit: 1,

      subintervals: 10,

      result: null,

      accept: true

    };

  }

**Inheritance**

Class SimsonsRule สืบทอด จาก React.Component เพื่อดึงคุณสมบัติ Component มาใช้

class SimpsonsRule extends Component {

**Encapsulation**

React, state และ lifecycle methods เป็นตัวอย่างของ Encapsulation. State ไม่ควรเปลี่ยนแปลงโดยตรงนอกเหนือจากเมธอดที่ React จัดหาไว้

 this.state = {

      funcString: 'x\*x',

      func: 'x\*x',

      lowerLimit: 0,

      upperLimit: 1,

      subintervals: 10,

      result: null,

      accept: true

    };

**Composition**

สร้าง Component Show\_graph เพื่อแสดง graph

  <Show\_graph func={this.state.func} x={this.state.result} y={this.evaluateFunction(this.state.func, this.state.result)}/>

**Class TrapzoidalRule**

**Constructor**

constructor(props) {

    super(props);

    this.state = {

      fx: '',

      func: '',

      a: 0,

      b: 0,

      n: 1,

      result: null,

      y: null,

      // gui

      fx\_gui: '',

      func\_gui: '',

      a\_gui: 0,

      b\_gui: 0,

      n\_gui: 1,

    };

  }

**Inheritance**

Class TrapzoidalRule สืบทอด จาก React.Component เพื่อดึงคุณสมบัติ Component มาใช้

class SimpsonsRule extends Component {

**Encapsulation**

React, state และ lifecycle methods เป็นตัวอย่างของ Encapsulation. State ไม่ควรเปลี่ยนแปลงโดยตรงนอกเหนือจากเมธอดที่ React จัดหาไว้

this.state = {

      fx: '',

      func: '',

      a: 0,

      b: 0,

      n: 1,

      result: null,

      y: null,

      // gui

      fx\_gui: '',

      func\_gui: '',

      a\_gui: 0,

      b\_gui: 0,

      n\_gui: 1,

    };

**Composition**

สร้าง Component Show\_graph เพื่อแสดง graph

<Show\_graph func={this.state.func\_gui} x={this.state.result} y={this.state.y}/>

**สรุป**

**ปัญหาที่พบระหว่างการพัฒนา**

1. การพัฒนา website ด้วย JavaScript โดยใช้ frame-work React ช่วงแรกในการศึกษา เขียน website ด้วย Function ทั้งหมด ทำให้ การปรับเปลี่ยนการเขียน แบบ Object-Oriented ต้องใช้เวลาพอสมควร ทำให้กำหนดการระยะเวลา ขยาย
2. Learning Curve สูง เนื่องจากการพัฒนา website จำเป็นต้องใช้ทักษะหลายด้าน ทำให้ ขอบเขตระยะเวลาทำโครงงาน ขยาย และ คุณภาพ website อาจไม่เป็นไปตามเป้าหมาย
3. เมื่อ ทำ Testing ทีละ ส่วนของ website พบข้อผิดพลาดบางส่วน

**จุดเด่นของโปรแกรม**

Object ต่างๆล้วนเป็นรูปแบบ Composition ชัดเจน