

ข้อ 1 - CPU,GPU,FPGA

สารบัญ

- phase1: disclaimer
- phase2: intro
- phase3: prepare database , install & run
- phase4: mvc concept
- phase5: requirement
- phase6: UI
- phase7: url
- phase8: อธิบายการทำงานของตัวโปรแกรม

[phase1: disclaimer]

- ผมได้ comment code ใส่ส่วน edit data ทิ้งไว้เนื่องจากทำไม่สำเร็จ ใน folder controller (ซึ่งตรงส่วนนี้เป็น feature เสริมที่จะทำ)
 - delete และ result แม้ว่าอาจารย์ไม่ได้สั่งใน requirement แต่ผมก็ใส่ไป (ทำนอกเหนือ requirement)
 - ถ้ารันใน safari จะไม่สามารถ click link ในหน้า home ได้ (เนื่องจากติดสิทธิ์เรื่องความปลอดภัยของฝั่ง client , cors ซึ่งถ้าลง ssl ก็จะทำรันได้ปกติ)
- แต่ถ้า run ใน google chrome ก็ใช้งาน web application ที่ทำไว้ได้ปกติ

[phase2: intro]

ต้องอธิบายก่อนว่า

ใช้ภาษา javascript เป็นภาษาหลักเพื่อเขียน web application โดยเฉพาะ โดยต้องติดตั้ง node js version LTS มาก่อนด้วย

express = เป็น framework สำหรับการทำ backend

json file = เป็น database ชั่วคราว

ejs = เป็นไลบรารีสำหรับการแสดงผล ui

- ในส่วนของการ ไล่อ่านไฟล์ ให้ผู้ใช้ได้จาก

server.js -> routes/route.js -> controller : endpoint.js -> logic.js -> operator.js -> ซึ่งหลังจากนี้จะตอบกลับเป็น .json หรือไปเรียกต่อที่ view/index.ejs หรือ view/results.ejs

[phase3: prepare database , install & run]

[prepare database]

- database ไม่ต้องลงเราใช้ผ่าน json ได้เลย

[install]

- ทำการติดตั้ง node_module หลังจากแตกไฟล์ .zip แล้วให้เข้าไปใน folder แล้วเปิด prompt รันคำสั่ง
“npm i” หลังจากที่เรารันคำสั่งนี้จะปรากฏ folder ชื่อว่า node_modules (ซึ่งถ้าถามว่ามันมาจากไหน มันไปโหลดติดตั้งจากไฟล์
package.json) [ติดตั้ง node ให้เรียบร้อย โดยแนะนำติดตั้งตัว LTS]

[run]

- ที่ prompt พิมพ์คำสั่ง “npm start” เพื่อทำการเปิด server โดยที่ prompt จะขึ้นข้อความว่า
‘[HOST] <http://localhost:3000> \n Database Connected !’ ซึ่งเป็นการบอกว่าเราเปิด server สำเร็จแล้ว ให้ไปที่ browser แล้ว
serch ว่า
<http://localhost:3000> จากนั้นจะเป็นปรากฏเป็นหน้า homepage ของเรา

[phase4: mvc concept]

- mvc -> model view controller เป็น แนวคิดหนึ่งในการสร้าง software โดย model จะเปรียบเสมือน data , view เปรียบ
เสมือน interface หรือ หน้า ui และสุดท้าย controller เป็นตัวกลางในการสื่อสารหรือส่งผ่านข้อมูลระหว่าง model และ view
- ในตัวโปรเจกต์นี้ ผมจะแบ่ง structure ตามหลักการนี้ดังนี้

```

|— README.md
|— controller
|— model
|— node_modules
|— package-lock.json
|— package.json
|— routes
|— server.js
|— view

```

server.js = main file สำหรับการรัน server นี้ คล้าย main.java

package.json = file ที่เอาไว้บอกว่าตัวโปรเจกต์เราจะใช้ library หรือ เครื่องอะไรบ้าง

routes/route.js = เป็นตัวบอกว่า software ที่จะสร้างมี url อะไรบ้าง

model = เป็น ‘โครง’ สำหรับการเตรียมข้อมูลที่จะใช้ใน server ถ้าไม่มีโครงก็เราก็ไม่รู้ data ที่จะรับและส่งมี field อะไรใดๆ
บ้าง

controller = เป็นตัวกลางที่สำคัญมากในโปรเจกต์นี้ โดยมันจะทำหน้าที่เปรียบเสมือนเป็นตัวกลางในเรียกหากระหว่าง model และ
view โดยภายใน controller จะประกอบไปด้วย

- endpoint.js = เป็นส่วนทำการรับ request ที่ส่งมาจาก client เช่น header และ ข้อมูลใน header จากนั้นจะส่งต่อไปที่ไฟล์
logic.js

- logic.js = เปรียบเสมือน สมอ สำหรับการ process และคำนวณต่างๆ โดยในส่วนนี้ผมได้สร้าง function compiler เพื่อให้ทำ

ตาม requirement อาจารย์ โดยเฉพาะ ซึ่งเมื่อทำงานเสร็จจะส่งต่อไปที่ operator.js

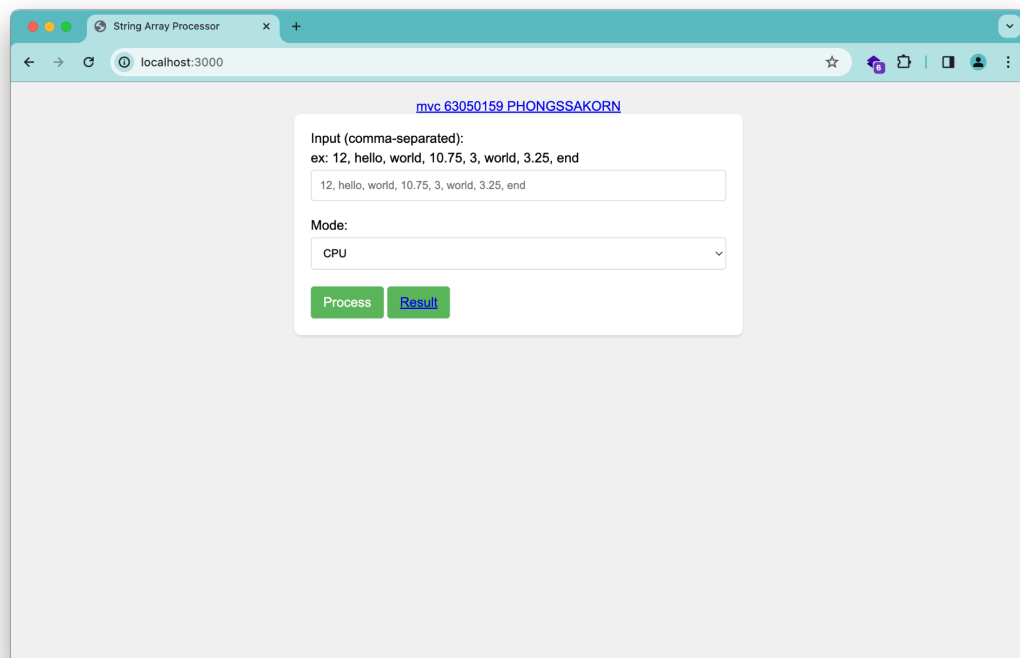
- operator.js = เป็นส่วนที่เตรียมพร้อมสำหรับยิง sql script เข้า database ซึ่งเมื่อเรายิงไปแล้วทาง database ก็จะส่ง ข้อมูลที่เราต้องการกลับมา เมื่อได้รับข้อมูลแล้ว ก็จะส่งเป็น return สุดท้ายไปหา client ผ่านตัวแปรที่ชื่อ res
view = เป็นส่วนสำหรับการแสดงผลหน้า ui ซึ่ง output สุดท้ายที่ client ได้รับถ้าเรียกผ่าน url ของ / หรือ /report จะตอบกลับเป็น .html file เสมอ โดยจะใช้ library ที่ชื่อว่า ejs ซึ่งการทำงานคล้าย html เลยแต่ว่ามันสามารถที่จะ process ต่างๆ โดยไม่ต้องใช้ javascript ช่วยเลย แต่จะใช้ผ่านไวยากรณ์ <% code %>
ซึ่งข้อดีคือไม่ต้องใช้ javascript ซับซ้อน

[phase5: requirement]

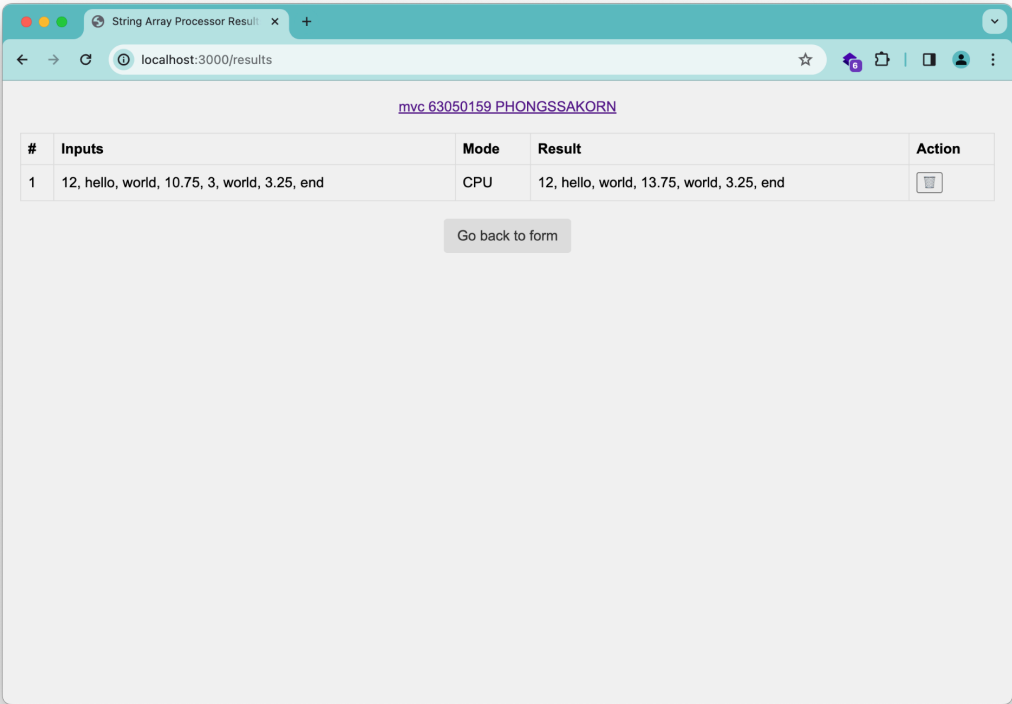
ในส่วนของ requirement โจทย์ที่อาจารย์ต้องการ ผมได้เขียนเป็น function ไว้ใช้งานชื่อว่า arrayFunctions อยู่ในไฟล์ controller/utis.js ซึ่งจะทำงานตาม scope ของ requirement ที่อาจารย์กำหนดไว้

[UI]

index.ejs



results.ejs



phase7: api

JavaScript

```
router.get('/', (req, res) => {  
  res.render('index', { result: null, error: null }); // Render the EJS  
  template with no initial data  
});  
  
router.post('/', new Endpoint().arrayProcessEndpoint);  
router.get('/results', new Endpoint().arrayResultEndpoint);  
  
router.post('/delete-index/:index', new  
Endpoint().arrayDeleteIndexEndpoint);
```

[get] / = หน้า home

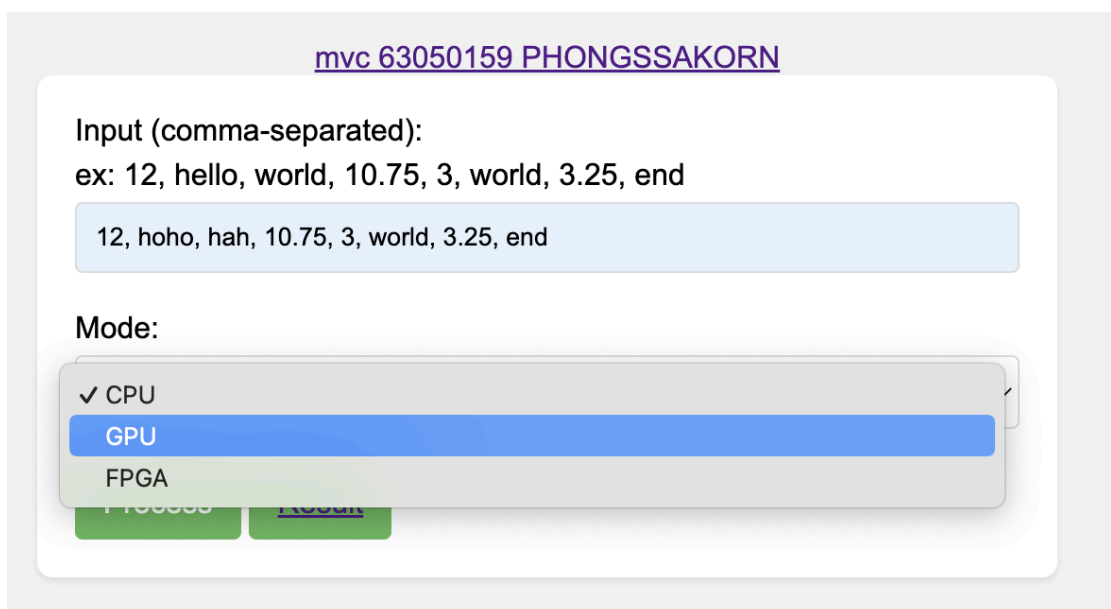
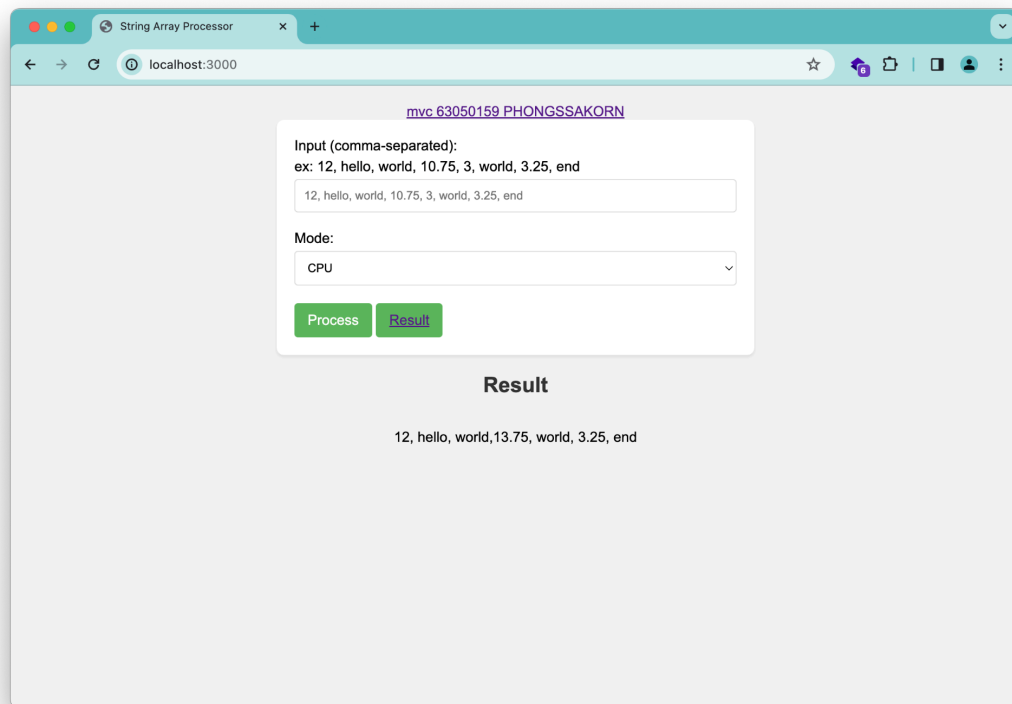
[get] / = หน้า ui หลัก หรือ home

[post] / = เส้นสำหรับการประมวลผลตัวอาร์เรย์ โดยรับข้อมูลเป็น ชุดข้อความ และ mode

[get] /result = สำหรับดูหน้า result

[post] /delete-index/:index = สำหรับการลบ ข้อมูลใน json จำนวน หนึ่งแถว ผ่าน id

[phase8: อธิบายการทำงานของตัวโปรแกรม]



1. กรอก array ของข้อความ จำนวน 10 ข้อความ โดยแต่ละข้อความต้องแยกด้วยเครื่องหมาย comma จากนั้นให้เลือกประเภทของตัวประมวลผล CPU, GPU หรือ VGA เมื่อเสร็จสิ้นแล้วให้กดปุ่ม Process จากนั้นโปรแกรมจะประมวลผลแล้วจะแสดงข้อมูล

ออกมาทางจอภาพ ที่บรรทัดใหม่ หรือ Result

#	Inputs	Mode	Result	Action
1	12, hello, world, 10.75, 3, world, 3.25, end	CPU	12, hello, world, 13.75, world, 3.25, end	
2	12, hello, world, 10.75, 3, world, 3.25, end	GPU	end, 3.25, world, 3, 10.75, worldhello, 12	
3	12, hello, world, 10.75, 3, world, 3.25, end	FPGA	10.75, world, hello, 12, end, 3.25, world, 3	

[Go back to form](#)

2. หน้า result ไว้ใช้สำหรับดูข้อมูลที่ถูกบันทึกลงในฐานความจำชั่วคราว (.json)

#	Inputs	Mode	Result	Action
1	12, hello, world, 10.75, 3, world, 3.25, end	CPU	12, hello, world, 13.75, world, 3.25, end	
2	12, hello, world, 10.75, 3, world, 3.25, end	GPU	end, 3.25, world, 3, 10.75, worldhello, 12	

[Go back to form](#)

3. หน้า result สามารถ เลือกลง บรรทัด ที่ต้องการจะลบออกได้