

[Open in app](#)

# Tangmo Pavat Lertpiromlak

[About](#)

## เมื่อฉันมาจับ Frontend—My Individual Study on Web Development



Tangmo Pavat Lertpiromlak · 1 hour ago · 7 min read

ประสบการณ์ในการทำ web application เกี่ยวกับการจัดการ event โดยใช้ React กับ Laravel



Photo by [dylan nolte](#) on [Unsplash](#)

สวัสดีค่า เราขี้อัดงโนนนะ เรียนอยู่ภาควิชคอม คณะวิศวกรรมศาสตร์ จุฬาฯ ตอนนี้ก็พึ่งจบปี 3 ก็เลยจะมาเล่าถึงประสบการณ์ในการทำ individual study โดยใน post นี้ก็จะเน้นเล่าเรื่องมากกว่าการเล่าสิ่งที่เป็น technical หรือการแปะ code ให้ดู

• • •

## Prologue

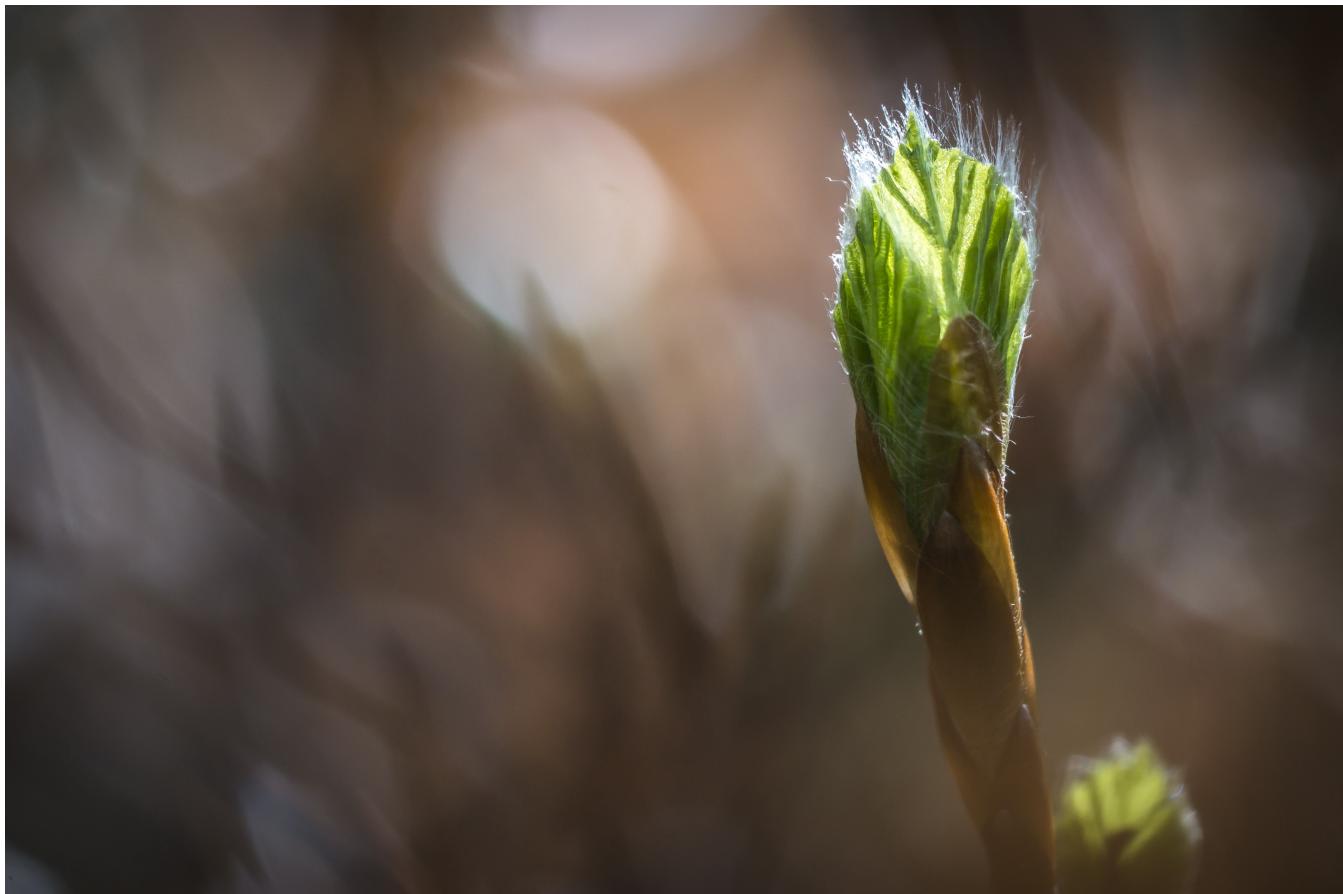


Photo by [Hansjörg Keller](#) on [Unsplash](#)

แน่นอนว่าจริง ๆ แล้ว เราภารกิจเหล่าว่ามันมีวิชา individual study แต่ด้วยความว่า (ขออ้าง) เราเป็นปี 2 เลือกภาค ก็เลยต้องเรียนวิชาเพิ่มเติมที่ปี 1 คอมตรงเรียนไปแล้ว ทำให้อาจจะไม่มีเวลา malangวิชา indiv แต่พอมาถึงปี 3 เทอมต้น รู้ด้วยว่าต้องทำให้กับตัวเองที่ต้องกลับมาทำงานแล้ว เราภารกิจเริ่มรู้สึกว่า ถึงเวลาต้อง up skill และล่ะ โดยเฉพาะ skill ด้าน frontend เพราะเราทำแต่ backend ตลอดมา เดียวจะไปสู่ full stack ไม่ได้นะ!

อีกเหตุผลหนึ่งคือมีเพื่อน ๆ มาช่วยทำ indiv เราภารกิจขอออดตามเพื่อนไปด้วย สุดท้ายก็มาจบที่ MyCourseVille กับ อ.โปรดปราน โดยมีพี่ ๆ ที่ MyCourseVille ช่วยดูแลการทำ project ของ indiv ด้วย โดยวิชา indiv ที่ทำนี้ จะสนใจเกี่ยวกับ web development เป็นหลัก โดยเป็น project เกิดใหม่ (இல்லை) ของ MyCourseVille ที่เป็นระบบการจัดการ event

## รู้อะไรก่อน?

อันดับแรกเลย คือต้องบอกว่าเราค่อนข้างโชคดีที่เคยใช้ Laravel มาหลายครั้งแล้ว เคยแตะภาษา PHP มา ก่อน เคยเขียน HTML, CSS มาก็บ้าง แต่พวกเทคโนโลยีฝั่ง frontend ที่ใช้ในการทำให้หน้าเว็บมัน interactive ที่เป็นของใหม่ ๆ ก็แทบไม่เคยแตะเลย อาจจะมีได้ลองใช้ Vue กับ React บ้าง (กับ jQuery ในตำนาน<sup>(๑)</sup>) แต่ก็ไม่ได้จริงจังขนาดนั้น

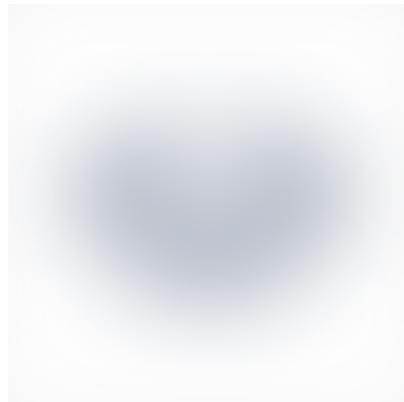
## เพื่อน ๆ

ในทีมที่ทำ event app มีกันทั้งหมดประมาณ 7 คน และมาเพิ่มอีก 2 คน ในตอนหลัง โดยที่ทุกคนก็เป็นเพื่อน ๆ เราในภาคคอมเหมือนกัน โดยจะแบ่งเป็น frontend ประมาณ 4 คน ที่เหลือเป็น backend ทั้งหมด

## เลือก #TeamFrontend

เนื่องจากยังไม่เคยทำ frontend แบบจริงจังมาก่อน ก็เลยเลือกที่จะทำตรงส่วนนี้ เพราะตัวเราเองก็มีประสบการณ์กับ backend มากบ้างแล้ว ก็เลยเลือกที่จะลอง explore ของใหม่ ๆ ลองเขียน code ของจริงกับ frontend เพื่อหาประสบการณ์ และเหมือนเป็นการ up skill ตัวเองให้เป็นระดับ full stack ไปในตัวด้วย

## อยากให้มา (Yak-Hai-Ma)



Logo ของเว็บ “อยากให้มา”

เป็น web application ที่เขียนโดยใช้ React + Laravel โดยเป็นเครื่องมือที่ใช้เชื่อมโยงระหว่างผู้จัดกิจกรรม (Event) กับผู้เข้าร่วมกิจกรรม ให้คิดง่าย ๆ มันก็คล้าย ๆ กับ Eventpop, Eventbrite นั่นแหล่ะ! ต่อจากนี้ เราขอเรียกมันย่อ ๆ ว่า event app แล้วกัน

## React + Laravel

คุณต้องปรับตัวให้เข้ากับบริษัทที่คุณทำงานอยู่ให้ได้ — อาจารย์นิรนามท่านหนึ่ง

ก็อย่างที่ว่า พึ่งเคยเห็นเหมือนกัน (หรือเราไม่ตามชานบ้านเองหว่า?) จับ Laravel มารวมกับ React อาจจะพะรำว่า พี่ ๆ และอาจารย์ที่อยู่ในทีม และระบบ MyCourseVille (เป็น LMS ที่ใช้

ภายในคณะวิศวฯ จุฬาฯ) ใช้ PHP กันมาก่อน และระบบที่ทาง MyCourseVille เดย์พัฒนามา ก่อนใช้ Laravel

## (นอกเรื่อง) COVID-19 🎉

เป็นเรื่องที่ unexpected จริง ๆ ตอนแรกนึกว่าจะได้มาเจอน้ำกัน พอดีน COVID ระลอกสอง ก็ต้องมาทำงานจากที่บ้าน พากเราก็ต้องคุยกันผ่าน Discord พ่อระลอกซองชาลงก่อนสอบ midterm ก็ได้มาเจอกันบ้าง แต่หลังจากนั้นไม่นาน ระลอกสามก็ต้นมาอีก แฉมหนักกว่าเดิม! กล้ายเป็นมา พากเราได้ประชุมเจอน้ำกันแค่ 2–3 ครั้งเอง ก็แบบเสียดายเหมือนกันนะ . . .

## Bingo and Reward (Dec 20 — Jan 21)

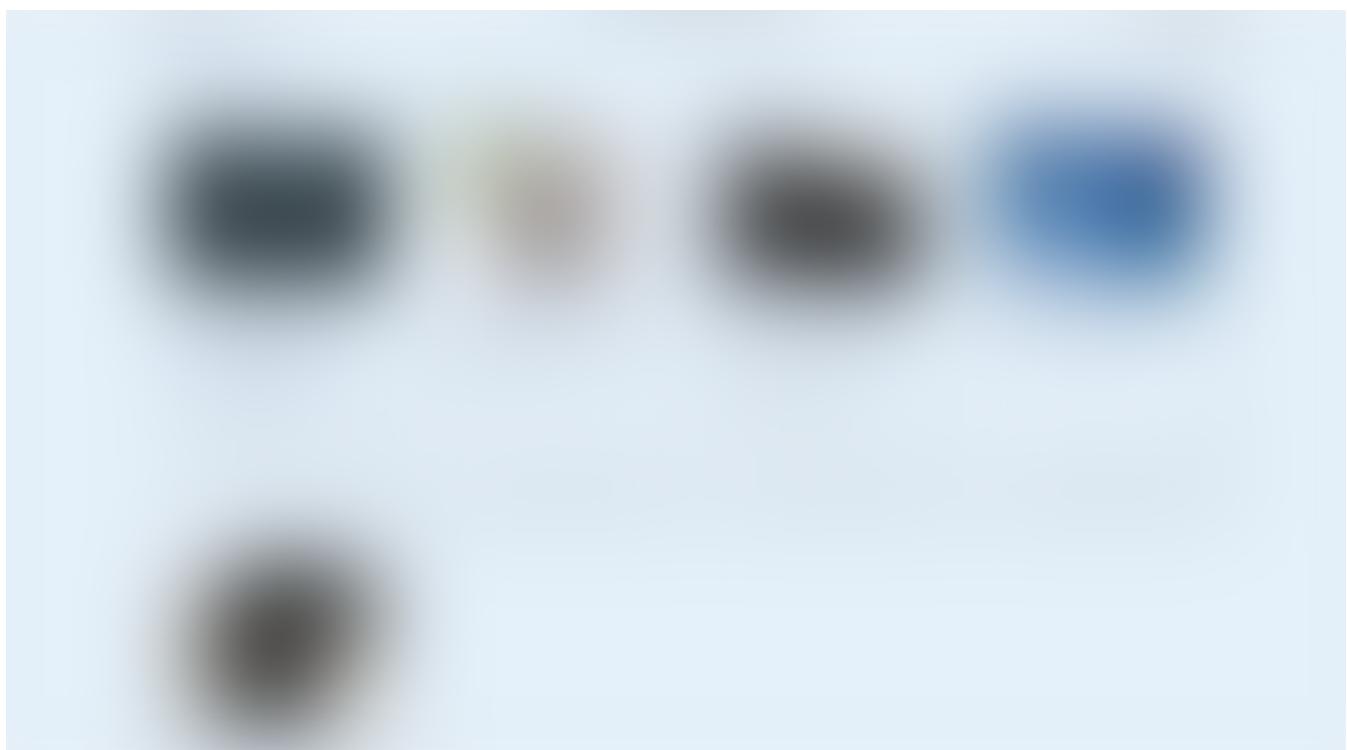
งานแรกที่เราได้รับมา คือให้ทำหน้าระบบรางวัล และรางวัล 1 ประเภท นั้นก็คือ bingo โดยที่พี่ ๆ และอาจารย์จะ provide ตัว styling และตัวอย่าง component บางส่วน เช่น ตาราง, list, card มาให้ ส่วนที่เหลือเราจะต้อง implement เอง หรือบางอย่างก็เป็น fully original component ไปเลย เช่น ตัวตาราง bingo

### อะไรคือ Bingo นะ?

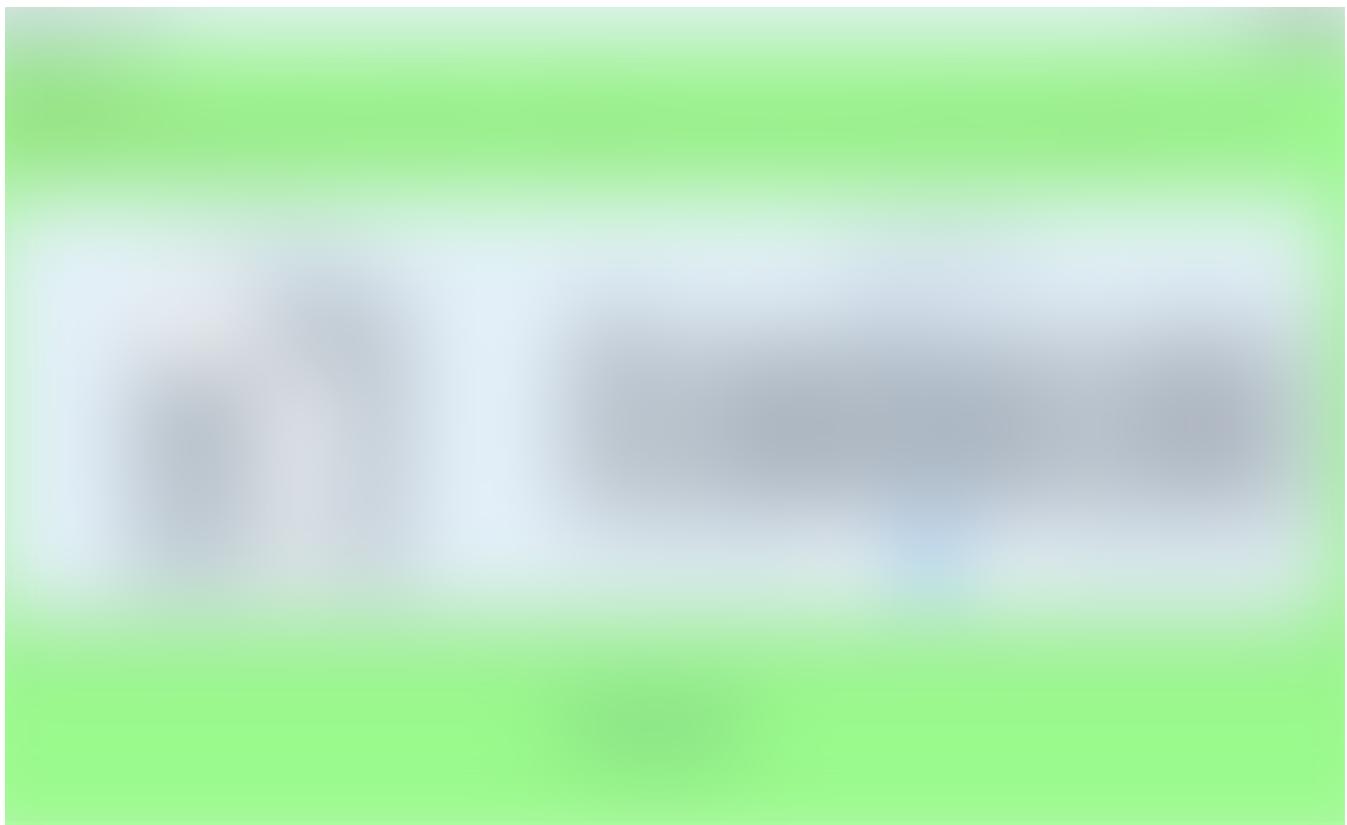
Bingo ก็คือกิจกรรมที่จะให้ผู้เข้าร่วม event เป็นคนร่วมสนุก โดยผู้เข้าร่วมแต่ละคนจะมีกระดาน bingo เป็นของตัวเอง 1 สำหรับ event นี้โดยเจ้าของ event จะเป็นคนสุ่มหมายเลขขึ้นมา

### ทำอะไรบ้าง

- หน้ารางวัลของฉัน — Card รางวัลแต่ละอัน, Pagination



- หน้า Bingo — ตาราง Bingo, ตารางประวัติการออก Bingo, Pagination



## My Workflow

หลัก ๆ เลย การสร้างแต่ละหน้าใน frontend ก็จะเป็นประมาณนี้ (ไม่รวมการ test)

1. Draft — เขียน HTML กับ CSS สำหรับทั้งหน้าขึ้นมาก่อน อาจจะทำตาม design ที่เพื่อนที่ทีมเขียนไว้ใน Figma หรือเขียนขึ้นมาสด ๆ เลย และปรับตาม design ที่หลัง แต่หลัก ๆ คือให้มีโครงขั้นมาก่อน ก่อนจะไปขั้นตอนถัดไป
2. Mock — เตรียม mock data ไว้ในขณะที่ทีม backend ยังทำ API ไม่เสร็จ ตรงจุด ๆ นี้โค้ดมันก็จะเละๆ หน่อย 😅
3. Refactor — ได้เวลาแล้วล่ะที่จะแยก component เป็นชิ้น ๆ
4. Integrate — ส่วนนี้เราจะทำการเชื่อม frontend ของเราเข้ากับ backend

มีอีกอย่าง ก็คือการกระโดดไปช่วยดู backend บ้างเป็นบางครั้ง ค่อยช่วยเพื่อน ๆ ที่เป็น newbie ด้าน PHP และ Laravel แต่ก็แค่อาจจะแนะนำอะไรเล็กน้อย ไม่ได้เข้าไปช่วยเขียน code อะไรมากนัก เพราะ backend มันก็ส่งผลกระทบถึง frontend เหมือนกันนะ ถ้า backend ดี frontend ก็ติดตามด้วย

## Git?

ที่ MyCourseVille เราไม่ใช่ GitHub กันนะ เราใช้ GitLab แทน แต่ดูรวม ๆ แล้ว ถ้าใช้แค่ตัว version control ความสามารถก็เท่านั้นไม่แตกต่างกัน แต่พี่ ๆ เค้าขยายมาว่า GitLab เจ๋งกว่า แต่ในตอนที่ทำงานจริง ก็ยังไม่ได้ใช้ความสามารถมันเต็มที่อยู่ดี

## เปิด Documentation รัว ๆ

สิ่งที่หนีไม่พ้นในการเขียน code ก็คือการเปิด documentation ของแต่ละ framework ขึ้นมาดู สิ่งที่เราเปิดบ่อย ๆ ก็คงจะเป็น Bootstrap เพราะว่าต่อให้เราใช้ React Bootstrap แต่ component สำเร็จรูปก็ไม่ได้ตอบโจทย์ทุกอย่าง โดยเฉพาะ custom component อย่าง Bingo ที่ทำให้เราต้องใช้ utility class รัว ๆ

## TypeScript!

TypeScript คือ superset ของ JavaScript โดยเป็นการเพิ่ม static typing ขึ้นมาแบบ optional (static type ก็คือตัวแปรมีประเภทตัวแปร แบบ C++, Java นั้นเอง)

จะบอกว่า TypeScript มี overhead เยอะ แต่เขียน ๆ ไปก็จะเห็นประโยชน์很多 โดยเฉพาะเวลาเขียน props ให้ component หรือ share code กับเพื่อน ๆ คนอื่น

แต่บางที่ IDE ช้า (เราใช้ VS Code) หรือคอมช้าก็จะหงุดหงิดหน่อย บางทีก็ highlight ตัวแดง ๆ โดยไม่มีเหตุผล พอผ่านไปแปปนึง ก็หายไปเลยเลย งมาก ๆ

## เขียน Bingo

สำหรับโครงสร้างข้อมูลใน component ก็จะเป็น array ช้อน array ช้อน array (สุด ๆ ไปเลยมั้ย ล่ะ) โดยเก็บเป็นแคลว เป็นหลัก เป็น array (น่าจะบอกว่าเป็น tuple มากกว่า) โดยเก็บตัวเลข bingo และสถานะว่าถูก bingo หรือไม่

ในส่วนของหน้าตา เนื่องจากเว็บตัวอย่างที่อาจารย์เคยทำไว้ให้ ที่เป็น static site ก็มี CSS ออย แล้ว เราเก็บลงมือแกะ CSS class name และเอาไปลงม้า ๆ ดู จนออกมารูปหน้าตาตามข้างบน โดยเราใช้ table layout ช้อนกับกล่อง (div) และ apply CSS class จนได้หน้าตาที่พอใจออกมานะ

พอได้หน้าตาแล้ว ต่อไปก็เป็นเรื่องของ responsive ก็คือหน้าเว็บเราต้องตอบสนองต่อขนาดหน้าจอที่ต่างกัน ซึ่งก็มักจะหมายถึงใช้กับโทรศัพท์มือถือได้นั่นเอง วิธีแบบง่าย ๆ ก็คือใช้ developer tools ของ web browser ในการจำลองขนาดของหน้าจอแต่ละ device เช่น โทรศัพท์, tablet, คอมจอล็อก, คอมจอยใหญ่ และลองดูว่าหน้าตามัน ok หรือไม่ ควรเปลี่ยนการแบ่ง column หรือไม่ หรือสำหรับหน้าจอหนึ่ง ควรย้าย component ไปต่อบรรทัดใหม่หรือไม่ พอปรับได้จังลงตัวแล้วก็เป็นอันเสร็จสิ้น

## ติดต่อกับ Backend

```
useEffect, axios.get, setState
```

ใช้แล้ว มีแค่นั้นแหล่ การเชื่อม backed ในยุคแรก ๆ (ที่ยังไม่โดนพี ๆ comment มา) ก็คือการส่ง HTTP request ไปหา backend เพื่อดึงข้อมูล bingo ออกมา โดยที่ library จะจัดการเรื่องการแปลงข้อมูลให้เป็น object โดยอัตโนมัติ เราไม่หน้าที่แค่เอาผลลัพธ์ของมันมาใช้ได้เลย แค่

ต้องระวังว่า บางที่ data structure อาจจะไม่เหมือนกันระหว่าง API doc กับของจริง หรือ convention ในการเขียนตัวแปร เช่น camel case กับ snake case ที่ใน database ไม่เหมือนกัน กับใน frontend

## State Management

พอถึงเวลา ก็ต้องย้าย state และ logic การ fetch หรือ modify ที่อยู่ใน component ไปไว้ในส่วนที่ share กันได้ ในที่นี้ event app ตัวโครงที่พี่ๆ ทำมาให้มี library MobX สำหรับทำ state management ติดมาให้อยู่แล้ว และก็ ตัว MobX ก็ถูกนำไปใช้ทำระบบ login ไปแล้ว ถ้าเราจะเปลี่ยนไปใช้ตัวอื่น ก็คงไม่น่าจะดีหรอกนะ

สำหรับ MobX จะบอกว่าตอนแรกไม่ค่อยเข้าใจเท่าไรว่ามันใช้อย่างไร พอดีดู tutorial ก็ยังไม่เข้าใจอยู่ดี ก็เลยลองพยายามด้วยการ copy code ตัวอย่างที่มีให้อยู่แล้วในระบบ login และมาดัดแปลงจนมันได้ แล้ววิเคราะห์กันว่าค่อยกลับไปดู tutorial อีกครั้ง

สรุป MobX ง่ายๆ คือเราประการ data class หรือ object มี getter, setter อะไรก็ว่าไป พอดี เราเปลี่ยนแปลง field ใน class นี้ หรือเรียก method ที่ทำให้ field เปลี่ยน ทุกๆ component ที่มี observer ครอบอยู่ ก็จะตอบสนองต่อการเปลี่ยนแปลงโดยอัตโนมัติ ก็คล้ายๆ กับยก react hook useState มาไว้ข้างนอก component นั้นแหละ

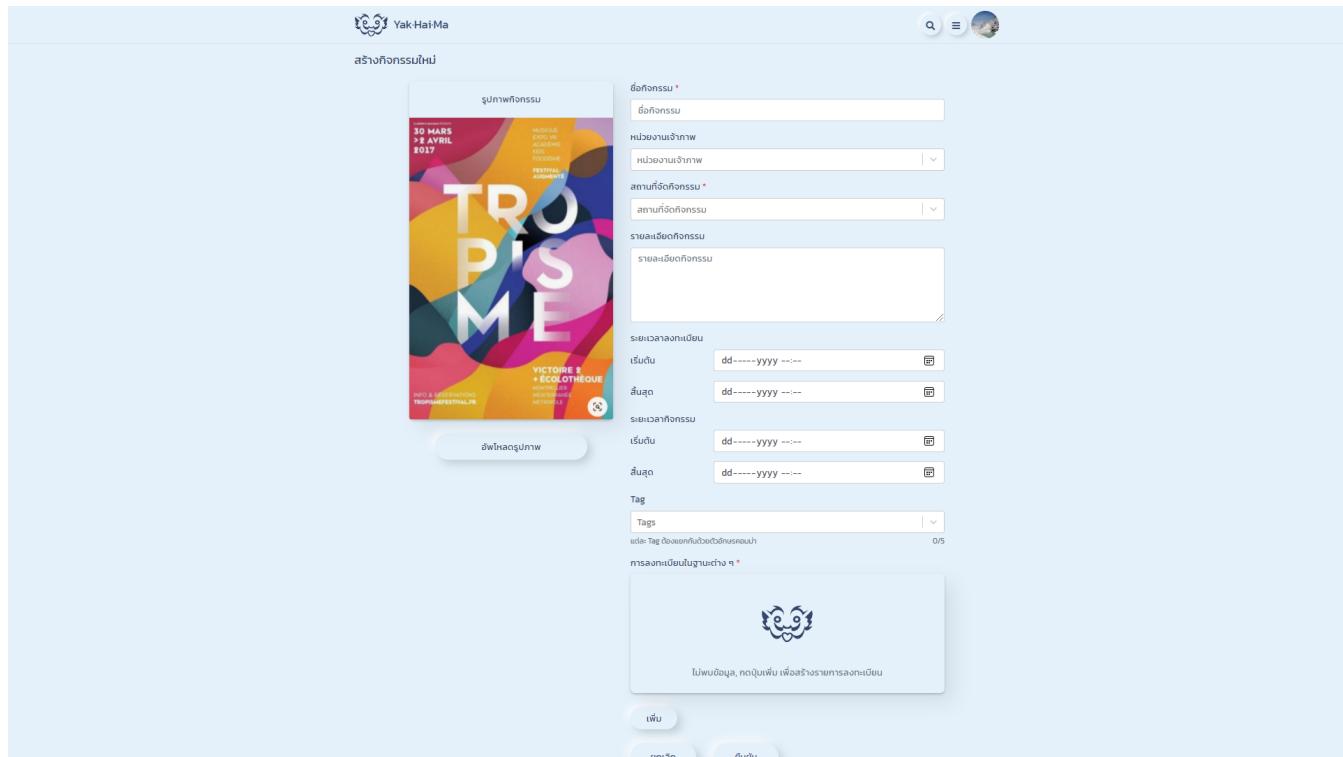
## ก่อนปิดงาน Bingo

แน่นอนว่าก็ต้องมีการ review โดยพี่ๆ และทุกคน ทุกคนก็จะเอางานมา demo ให้ดู และอาจจะเปิด code ประกอบด้วย พอดีการ review ก็อาจจะมีการแก้ code หรือแก้ function อะไรต่างๆ ก่อนจะ merge เข้า branch หลัก

• • •

## Event (Jan 21 — March 21)

พอดีงาน bingo ต่อไปก็เป็น main course นั้นก็คือตัว event นั้นเอง โดยเราเองได้รับผิดชอบในการช่วยทำหน้าสร้าง event ใหม่ (host an event) แต่จริงๆ แล้วก็ไม่ค่อยได้ทำอะไรมาก นอกจากสร้าง pure HTML/CSS และให้เพื่อนในทีม frontend ไปแปลงเป็น component และอาจจะมีคอย(บ่น) แนะนำการเขียน component การยก state และอื่นๆ อีกเล็กน้อยให้กับเพื่อนที่เขียน คอยตามแก้ bug และ optimize code บ้าง แต่เราไม่ใช่คนเขียน code หลัก (เอบรู๊สิก ผิดนนะ ที่ไม่ได้ช่วยอะไreyo เท่าไร 😅)



รูปหน้าสร้าง event ใน Figma

## หลักทางให้ Figma

ใน feature นี้ สิ่งแรกที่เราจะเริ่มทำ คือการ design ตัวระบบก่อนลงมือเขียน code สำหรับ frontend เราจะใช้เครื่องมือที่เรียกว่า Figma เป็นโปรแกรมออกแบบส่วนติดต่อผู้ใช้งาน (user interface) ตอนแรก เพื่อนที่ทำ UI ก็ทำมีมาตั้งนาน เป็น wireframe ขาดๆ ผ่านมาสักพัก พี่มาบอกใบ้ ถึงรู้ว่ามันมี template ของ Figma ที่ตรงกับแบบที่อาจารย์ทำเป็นตัวอย่างให้ดู

## 100% HTML/CSS

ในขั้นตอนการขึ้นโครง เราเป็นคนทำเองเกือบทุกด้าน วิธีการก็คล้าย ๆ กับ part ที่แล้ว เพียงแต่ว่าครั้งนี้ เรายield ตามแบบของ Figma เป็นหลัก ไม่มี F12 ช่วยแล้วนะ!

1. ขึ้นโครง row, column ก่อน
2. เขียน card สำหรับ upload รูปภาพ
3. เขียน input field — มี 3 แบบ คือ แบบธรรมดา copy มาจาก bootstrap ได้เลย, แบบ datetime copy และมาดัดแปลง ถ้าจำไม่ผิด น่าจะใช้ flex ช่วย, แบบ text area
4. เขียน ticket pool คร่าว ๆ คือเป็นฐานะของการลงทะเบียนใน event
5. เดินปุ่มต่าง ๆ ให้เรียบร้อย
6. Check responsive ลองลาก ๆ หรือเปลี่ยนขนาดหน้าจอตุ้นๆ ไม่ ok ก็ปรับแก้อย่างเช่น breakpoint ไปเรื่อย ๆ

## ได้เวลาแปลงร่าง

ต่อไปก็เป็นการแปลงจาก HTML, CSS เป็นๆ ให้เป็น component โดยขั้นตอนนี้เพื่อนเราเป็นคนทำทั้งหมดเลย เราแค่ช่วยนั่งดู code เลยๆ

ถ้าให้เรียนความยากง่ายในการแปลง ตัวที่ยากที่สุดคือ Ticket pool เพราะเราต้องทำทั้ง card และ modal ซึ่งข้างใน modal ก็มี form อีกหนึ่งด้วย แต่ตัวรายการยังต้องทำให้ drag and drop ได้อีก แต่โชคดีที่เรามี library ช่วย (React beautiful dnd จาก Atlassian เจ้าที่ทำ Jira นั้นแหล่ะ) ก็เลยทุนแรงไปได้ระดับนึง



ตัวอย่าง ticket pool

ความยากของการแปลง นั้นก็คือการ abstract สิ่งที่เหมือนกัน ให้เป็น component เดียวกัน และ การสร้างความหลากหลาย โดยเราต้องคิดว่า อะไรเหมือน อะไรต่าง สิ่งที่เหมือน ก็ให้เป็น static content ใน component อะไรต่าง ก็ให้ customize ผ่าน props ของ component และการคิด props กับการส่งต่อ state นี้แหล่ะ ที่เป็นสิ่งที่ทำหายที่สุดในการทำ custom component ที่ต้องยังไม่นับว่าต้องเขียน type definition ของ Typescript อีกนะ

## **Yup! that tasted Formik**

Formik ก็คือ library สำหรับการทำ form มี validation ให้ด้วย เมื่อใช้กับ Yup การทำ validation จะง่ายขึ้นมาก? ก็ไม่ง่ายนะ ถ้าข้อมูลของเรามี custom data เยอะๆ ที่ built-in rule ไม่มีให้ โดยในหน้านี้ เราใช้ Formik มาช่วยในการ manage state ของ form แทนที่เราจะต้องมาทำ pattern เดิมๆ คือการตั้ง state และใช้ props onChange กับ value ช่วยแบบใน tutorial ของ React

## **ทำไมเราไม่เลือก React select**

แหล่งเพื่อนเราก็นั่งเขียนตัว tag input ตั้งนาน ไอเรก์ไปช่วยหา demo จาก Vue-bootstrap มาปรับ ๆ จนมั่วอกมาได้อยู่ดี ๆ พิถึงมาแนะนำว่า “ทำไมพากเราไม่ใช้ react-select ล่ะ”

อาจจริง ๆ React select มันก็ใช้ง่ายนะ แต่ที่ยากก็คือการ customize ตัว style ของ input แต่เนื่องจากว่า style มันก็คล้าย ๆ ของ Bootstrap อยู่แล้ว เราก็เลยไม่ได้จำเป็นต้องปรับอะไรมาก

ยังไม่เสร็จ

อีกไม่นาน ก็จะใกล้สอบ midterm แล้ว แต่ host an event ยังไม่จบ เมื่อเราพึ่งมาพบว่า backend ไม่ได้ออกแบบมาตรฐาน สำหรับการสร้าง event! ในขณะที่ทีม backend กำลังวางแผน แต่ก็ไม่ทันแล้วสินะ งั้นโยกส่วนติดต่อ backend ไปทำที่หลังแล้วกัน

· · ·

## Interlude

พอ feature event ใกล้จะเสร็จ ก็มาถึงช่วงเทศกาลการสอบ midterm และหลังจากสอบ midterm เสร็จเราก็ไม่ได้กระโดดไปทำ event app ต่อโดยทันที เพราะยังมี project แทนการสอบ midterm ของวิชาหนึ่ง

แต่หลังจากสอบ midterm “ไม่กี่วัน ก็ถึงเวลา official launch ของ “อยากให้มา” และล่ะ ความตื่นเต้นก็มาเยือนกับทีมพี่ ๆ ที่เป็นคนรับผิดชอบการ deploy และการ demo ของจริงให้แขก กับกิจกรรมแรกที่จัดโดยคณะวิศวะของเรานั่นเอง

ก็เหมือนจะผ่านไปได้ด้วยดีเกือบทุกอย่าง ยกเว้นอย่างหนึ่ง คือเราลืมลบ mock data ในหน้า description ของ reward ทำให้รายละเอียดจาก database ไม่ขึ้น แต่โชคดี (๔๕) ที่เหมือนจะไม่กระทบ functionality ของระบบ ทำให้การ demo ของจริงในวันนั้นก็เหมือนจะผ่านไปได้ด้วยดี

หลักจาก launch event ก็มีการเก็บงานอีกเล็กน้อยในหน้า host an event จนเสร็จแล้วส่ง pull request ไปได้

แต่ไม่ทันไร งานกลุ่มนี้เริ่มมารุมเร้าจากทุกสารทิศ...

· · ·

## More? (March 21 — May 21)



Photo by [Alex Machado](#) on [Unsplash](#)

พอกบไป 2 features ที่พวกราได้พัฒนามันขึ้นมา พี ๆ ก็เสนองานมาหลายอย่างให้พวกราได้เลือกทำ อย่างเช่น

- เพิ่มเติมหน้า event เช่น search event
- หน้า profile
- ทำ automated testing
- ทำระบบ deployment and operation
- อื่น ๆ แล้วแต่สนใจ

Assign งานกันติดบดี แต่สุดท้ายแล้วก็...

## ไม่เสร็จหมด

จริง ๆ คือเราก็เลือกจะไปดูส่วนของ deployment and operation แต่ก็ความว่าได้ไปศึกษาอะไรเพิ่มหรือเปล่า ก็ตอบว่าได้นะ แต่ไม่ได้เอาไปใช้กับ individual study นะสิ เอาไปใช้ทำ project วิชาหนึ่ง (ที่งานอย่างเยอะ)

## Container และผองเพื่อน

เรารักไปศึกษาเรื่องนี้มา ซึ่งก็คือ Docker กับ Kubernetes (พอดี project วิชานั้นบังคับให้ต้องไปดู) สำหรับ docker มันก็คือเหมือนกับ virtual machine ย่อส่วนแหล่ง เพียงแต่ว่ามันเบากว่ามาก ข้อดีของการมี container คือมันทำให้ deploy ไปได้ทุกที่แล้วยังทำงานได้เหมือนกัน โดยไม่ต้องมา configure หรือลงโปรแกรมในเครื่อง server เอง และก็ยังเปิดโอกาสให้เครื่องมือต่าง ๆ มาช่วยในการจัดการฝูงของ container เพื่อทำสิ่งต่าง ๆ เช่นการทำ auto scaling ได้ เครื่องมือดังกล่าวก็เช่น Kubernetes เป็นตัว container orchestrator ที่เป็น open source และน่าจะดังที่สุดในตอนนี้ สำหรับ flow ของการทำ app ของเราให้เป็น container app ก็น่าจะประมาณนี้

1. เขียน Dockerfile สำหรับ app แยก frontend backend
2. Build image ขึ้นมา แล้วลองรัน
3. เอา image ขึ้นไปฝากไว้ใน container registry เช่น Dockerhub
4. เขียน resource definition (deployment, service, secret, configmap, ...etc)
5. ลองนำจากข้อ 4 ไปรันใน local cluster ดูก่อน ตัวที่คนแนะนำก็น่าจะเป็น Minikube
6. พอลองรันใน local cluster และ ก็ปรับ resource definition ให้เข้ากับ cluster ที่เราจะไป deploy ถ้าใช้ cloud และต้องการ features อะไรต่าง ๆ ที่มีเฉพาะ cloud เท่านั้น ก็อาจจะต้องสร้าง resource เพิ่ม เช่น ssl certificate หรือ static ip
7. ลองทดสอบ แล้วก็เป็นการเสร็จสิ้นการ deploy

แบบเสียดายเหมือนกัน ที่สุดท้ายก็ไม่มีเวลาได้ลองใช้ความรู้เหล่านี้

• • •

## Epilogue

มาถึงเดือนพฤษภาคม หลังสอบทุกอย่างเสร็จแล้ว clear project ทุกอย่างเสร็จแล้ว ก็จะเป็นการสรุปงาน event app และวิชา individual study โดยรวม (ผ่าน Discord จ้า เพราะ โควิดระบาดหนัก และมหาวิทยาลัยยังปิดอยู่) และก็จะเป็นการปิด course อย่างเป็นทางการ

ก็จบไปแล้ว สำหรับการทำ individual study เราขอสรุปรวม ๆ สิ่งที่ได้ประสบการณ์จากการทำ individual study ก็คือ

- ความรู้ React แบบเข้มข้น — จริง ๆ ได้เรียนรู้การเขียน React แบบเอาไปใช้จริง ได้ลองสร้าง component ใหม่ ๆ ที่มีความเป็น original ขึ้นมาด้วย (อย่างเช่น Bingo) รวมถึงการได้ลองเอา library ใหม่ ๆ มาใช้ด้วย

- เพิ่มประสิทธิภาพการใช้ Bootstrap และเพิ่มความรู้ CSS หลาย ๆ อย่างเลย ได้ลองเล่นกับระบบ grid, flexbox มากขึ้น จากแต่เดิมไม่ค่อยมีความรู้ตรงนี้เท่าไร
- ได้ไปช่วยดูเพื่อนทำ backend โดยใช้ Laravel และก็ได้เรียนรู้เทคนิคเพิ่มเติมในการทำ backend จากพี่ด้วย (อย่างเช่นเรื่องการเขียน service, การออกแบบ API สำหรับสร้าง resource)
- การสื่อสารและการสอนเพื่อนเขียน code ในยามโควิด
- (Bonus) ความรู้ในช่วงแรก และการทำ event app ในช่วงแรก ช่วยในเรื่องการสัมภาษณ์ฝึกงานได้เยอะเลย

สุดท้ายแล้ว เราก็ขอขอบคุณทุกคนเลย ที่ทำให้ได้รับประสบการณ์ดี ๆ ความรู้จากการทำ individual study ในครั้งนี้ ก็จะเป็นประโยชน์อย่างมากกับตัวเราในอนาคต เช่นกัน รวมถึงเพื่อน ๆ ทุก ๆ คนด้วย

*End of line*

Some rights reserved 

