

เข้มกำราธรวน JavaScript

การพัฒนา Full Stack ด้วย **React.js** และ
Node.js: Bootcamp ใน 10 มื้อ

แต่งโดย: บุนพ่อน โภสธา

กวดแท้: แสงลัดสะมี จันทะมินาวง

พากวิชาชีวะกำຄອມພິວເຕີ ແລະ ຕັກໂນໂລຊີຂໍ້ມູນຂ່າວສານ
ຄະນະວິສະວະກຳສາດ, ມະຫາວິທະຍາໄລແຫ່ງຊາດ

2025

ເນື້ອໃນ

ມີ້ທີ 1: ພຶ້ນຖານ Full Stack & Node.js

ມີ້ທີ 2: Express.js API ແລະ ຫຼັກການ REST

ມີ້ທີ 3: ການເຊື່ອມໂຍງຖານຂໍ້ມູນ - MongoDB ແລະ Mongoose

ມີ້ທີ 4: ການຈັດການຜູ້ໃຊ້ ແລະ ການຢືນຢັນຕົວຕົນ

ມີ້ທີ 5: ການພັດທະນາ Frontend - Interface ແລະ Styling

ມີ້ທີ 6: Tailwind CSS ແລະ Component Styling

ມີ້ທີ 7: ການເຊື່ອມຕໍ່ React ກັບ Backend API ຂອງທ່ານ

ມີ້ທີ 8: ການໂຕຕອບກັບຜູ້ໃຊ້ & dynamic features

ມີ້ທີ 9: ປັດຝຸນ ແລະ ປະສົບການຜູ້ໃຊ້

ມີ້ທີ 10: ໂຄງການສຸດທ້າຍ ແລະ ການຮຽນຮູ້ໃນອະນາຄົດ

ก้าวผัดทะมา Full Stack ด้วย React.js และ Node.js: Bootcamp ใน 10 มื้อ

Day 1: พื้นฐาน Full Stack &
Node.js

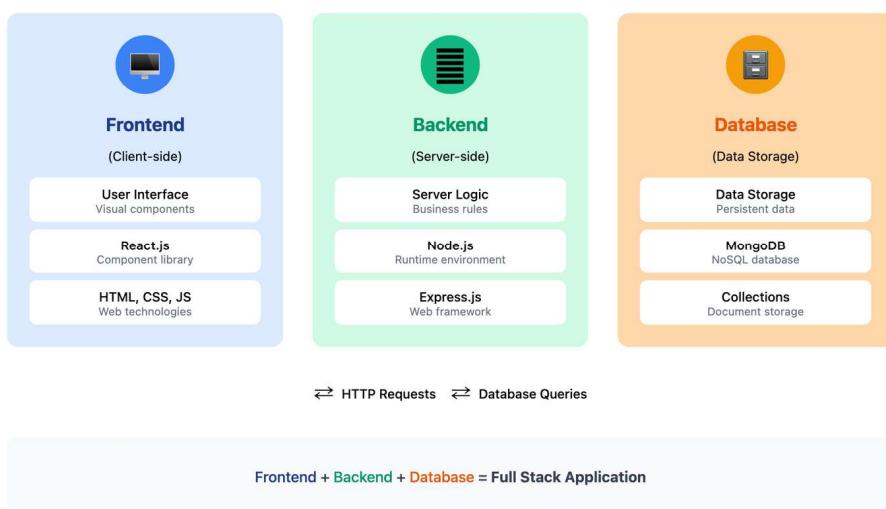
ပုဂ္ဂနာ

ຈຸດປະສົງ

- 1 ບົດສອນນີ້ແມ່ນມຸງເນັ້ນໃສ່ການປະຕິບັດ ແລະ ປະຍຸກໃຊ້ຈິງ
- 2 ທຸກໆຢ່າງທີ່ໄດ້ຮົນແມ່ນຈະສາມາດນຳໃຊ້ໄດ້ໃນ Final Project
- 3 ຈະໄດ້ຮົນຮູ້ເຕັກນິກຕ່າງໆ
- 4 ສາມາດຮົນໄດ້ຕະຫຼອດເວລາ

Full Stack Development ແມ່ນຫຍັງ?

Full Stack Architecture



Full-stack development ໝາຍເຖິງການສ້າງ web applications ສົມບູນແບບເຊື່ອຈະກອບດ້ວຍໜ້າຄາເວັບໄຊ (frontend), ການເຮັດວຽກຂອງຫຼັງບ້ານ (backend), ແລະ ການເກັບຂໍ້ມູນ (database).

ທັກສະຂອງນັກພັດທະນາ Fullstack

ນັກພັດທະນາ full stack ແມ່ນຈະຕ້ອງເປັນຄົນທີ່ມີຄວາມຮູ້ຢ່າງເລີກເຊິ່ງທັງ Frontend ແລະ Backend ລວມທັງເຄື່ອງມີອື່ນໆທີ່ຈໍາເປັນໃນການພັດທະນາເວັບແອັບພລິເຄົ້ນຈາກສູນຈົນເຖິງສໍາເລັດ



ທັກສະຂອງນັກພັດທະນາ Fullstack

ຄຸນສົມບັດ

- 🎨 Frontend ເຕັກໂນໂລຊີ: HTML5, CSS3, JavaScript ES6+, React.js
- ⚙️ Backend ເຕັກໂນໂລຊີ: Node.js, Express.js, RESTful APIs
- 🗄️ ການຈັດການຖານຂໍ້ມູນ: MongoDB, Mongoose, data design
- 🔧 ເຄື່ອງມີທີ່ໃຊ້ໃນການພັດທະນາ: Git, VS Code, npm, debugging tools
- 🌐 Deployment & DevOps: Cloud platforms, CI/CD, environment setup

ການຕັ້ງຄ່າສະພາບແວດລ້ອມການພັດທະນາ

ກ່ຽວກັບ NodeJS ແລະ ການຕິດຕັ້ງ Development Environment

Node.js ເປັນ runtime environment ທີ່ໃຊ້ງານໄດ້ຫຼາກຫຼາຍ, ເປັນ open-source ທີ່ຂ່ວຍໃຫ້ທ່ານສາມາດນຳໃຊ້ໂຄດ JavaScript ຢູ່ນອກເວັບບຮາວເຊີໄດ້. ມັນຖືກສ້າງຂຶ້ນເທິງ Chrome's V8 JavaScript engine ແລະ ຖືກອອກແບບມາເນື້ອສ້າງ ແລ້ວພລິເຄື່ອນເຄື່ອຂ່າຍທີ່ສາມາດປັບຂະຫຍາຍໄດ້.



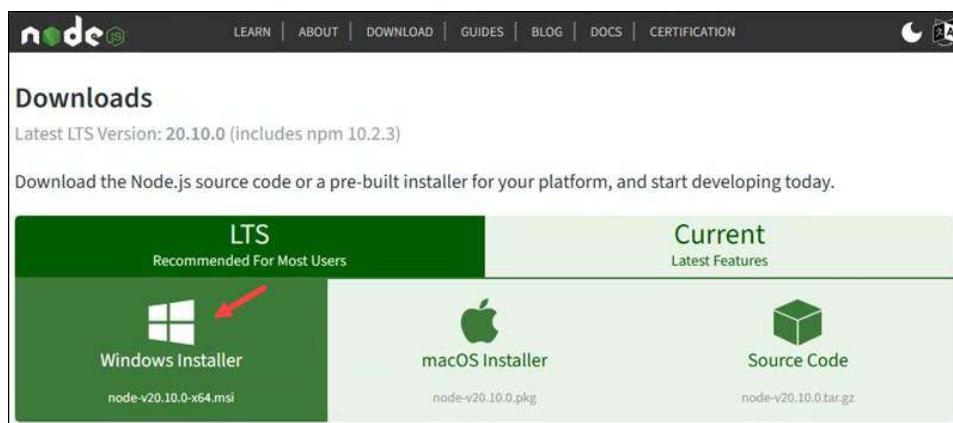
- ຕິດຕັ້ງ Node.js ແລະ npm (Node Package Manager):
<https://nodejs.org/en> (ແນະນຳໃຫ້ເລືອກ LTS Version)

ການຕັ້ງຄ່າສະພາບແວດລ້ອມການພັດທະນາ

ສໍາລັບ window

ຂັ້ນຕອນທີ 1: ດາວໂຫລດ Node.js Installer

ໃນເວັບບຮາວເຊີ, ໃຫ້ໄປທີ່ໜ້າ Node.js Downloads. ກົດປຸ່ມ Windows Installer ເພື່ອດາວໂຫລດເວີຊັນລ່າສຸດທີ່ໝັ້ນຄົງພ້ອມກັບ ການຮອງຮັບໄລຍະຍາວ (LTS). ຕົວຕິດຕັ້ງຍັງມີ NPM package manager ນຳອີກ.

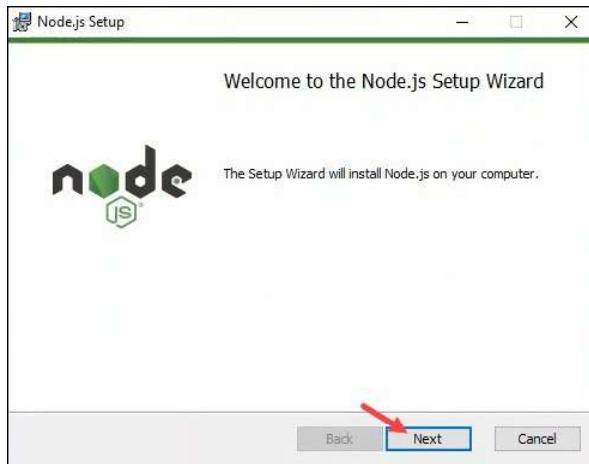


ການຕັ້ງຄ່າສະພາບແວດລ້ອມການພັດທະນາ

ຂັ້ນຕອນທີ 2: ຕິດຕັ້ງ Node.js ແລະ NPM

ຫຼັງຈາກດາວໂຫລດດີວຕິດຕັ້ງແລ້ວ, ໃຫ້ປະຕິບັດຕາມຂັ້ນຕອນລຸ່ມນີ້:

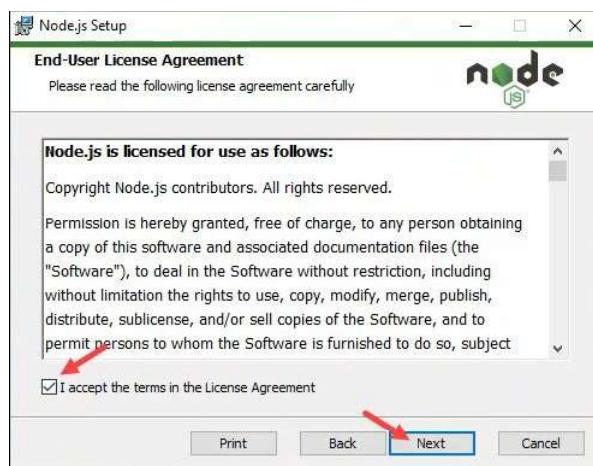
- ເລີ່ມດົວຕິດຕັ້ງໂດຍການຄລິກສອງເພື່ອໃສ່ໄຟລ໌ທີ່ດາວໂຫລດມາ.
- ດີວຊ່ວຍສ້າງການຕັ້ງຄ່າຂອງ Node.js ຈະເປີດຂຶ້ນພ້ອມກັບໜ້າຈໍຍິນດີຕ້ອນຮັບ.



ກົດ Next ເພື່ອສືບຕໍ່.

ການຕັ້ງຄ່າສະພາບແວດລ້ອມການພັດທະນາ

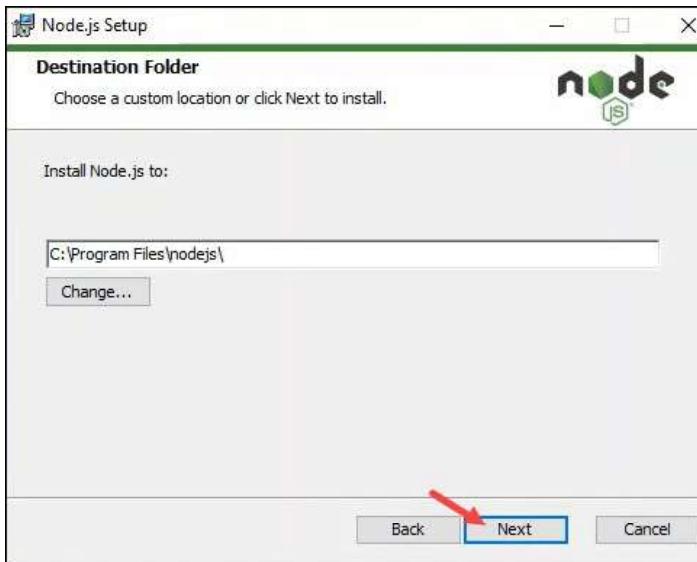
3. ທີ່ບໍ່ທວນຂໍຕົກລົງໃບອະນຸຍາດສໍາລັບຜູ້ໃຊ້ ແລະ ກົດໃສ່ຫ້ອງສີ່ຫຼຸຽມເພື່ອຍອມຮັບຂໍກໍານົດ ແລະ ເງື່ອນໄຂ.



ກົດ Next ເພື່ອສືບຕໍ່.

งานตั้งค่าสภาพแวดล้อมการพัฒนา

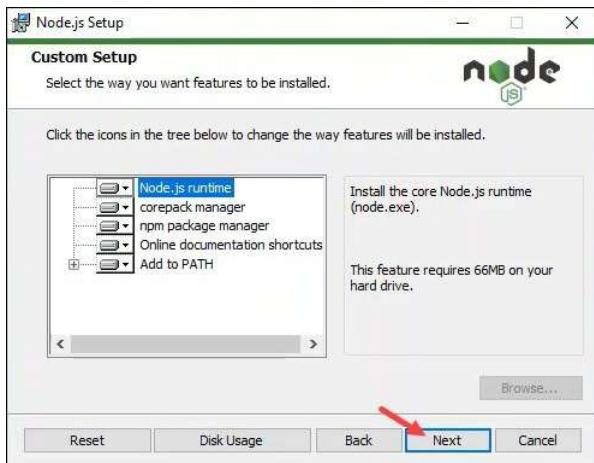
4 ต้องติดตั้งจะตามให้เลือกบ่อนติดตั้ง.



รักษาสถานที่ตั้งค่าเลื่อมตื้นสำลับ
งานติดตั้งแบบมาตรฐาน และ
กด Next เพื่อดำเนินงานต่อ.

งานตั้งค่าสภาพแวดล้อมการพัฒนา

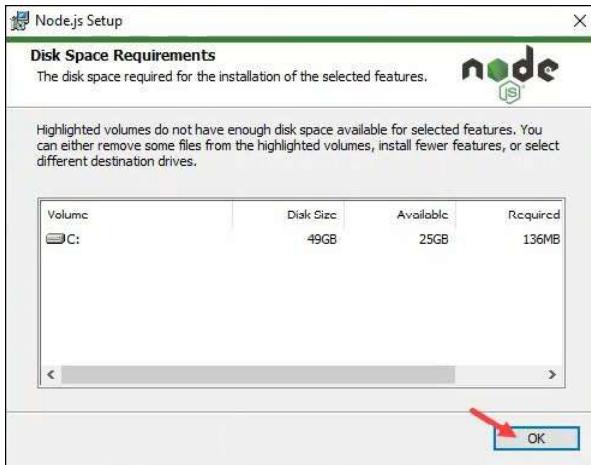
5. เลือกส่วนประกอบที่จะล้อมเข้า ทู ยิ่งเว้นออกจากงานติดตั้ง. ต้องเลือกตั้งค่าเลื่อมตื้นจะติดตั้ง Node.js, NPM, corepack, ทางลัดเอกสารสามของลาย, และเพิ่มโปรแกรมเข้าไปใน PATH.



ปับแต่งงานตั้งค่า ทู กด Next
เพื่อยอมรับค่าเลื่อมตื้น.

ການຕັ້ງຄ່າສະພາບແວດລ້ອມການພັດທະນາ

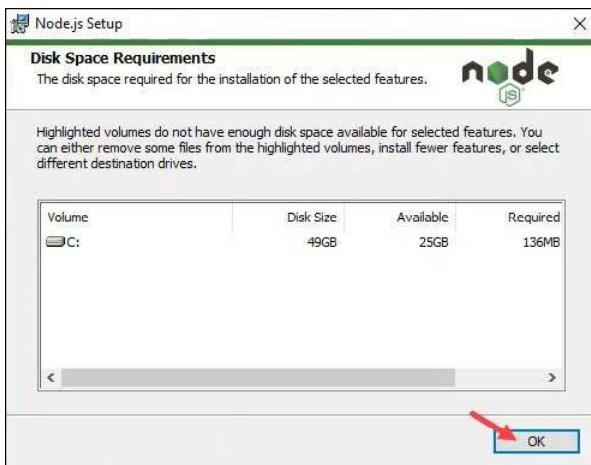
6. ພາກຕໍ່ໄປຈະສະແດງພື້ນທີ່ທັງໝົດທີ່ຈໍາເປັນສໍາລັບການຕິດຕັ້ງ ແລະ ພື້ນທີ່ວ່າງທີ່ເຫຼືອໃນດິສກ໌.



ກົດ OK ເພື່ອສືບຕໍ່. ຖ້າການຕິດຕັ້ງບໍ່ສາມາດດຳເນີນການຕໍ່ໄດ້, ໃຫ້ເລືອກດິສກ໌ອື່ນ ຫຼື ຕິດຕັ້ງຄຸນສົມບັດໜ້ອຍລົງ.

ການຕັ້ງຄ່າສະພາບແວດລ້ອມການພັດທະນາ

7. ເລືອກວ່າຈະຕິດຕັ້ງ dependencies ເພີ່ມເຕີມເພື່ອ compile native modules ຫຼືບໍ່. ບາງໂມຄູນ NPM ຈະຖືກ compile ຈາກ C/C++ ແລະ ຕ້ອງການເຄື່ອງມີເພີ່ມເຕີມເພື່ອເຮັດວຽກຢ່າງຖືກຕ້ອງ (ເຊັ່ນ: Python, Visual Studio Build Tools, ແລະ Chocolatey).

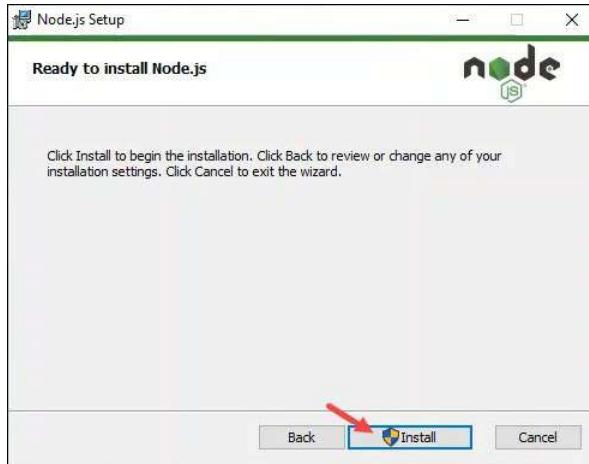


ຖ້າທ່ານໃຊ້ໂມຄູນດັ່ງກ່າວ, ໃຫ້ເລືອກກ່ອງສີ່ງໆ ແລະ ກົດ Next. ການເລືອກດົວເລືອກນີ້ຈະເລີ່ມ script ການຕິດຕັ້ງຫຼັງຈາກການຕິດຕັ້ງ Node.js ສໍາເລັດ.

ສໍາລັບການຕິດຕັ້ງແບບວ່າຍຕາຍ, ໃຫ້ຂ້າມຂັ້ນຕອນນີ້ ແລະ ກົດ Next ເພື່ອສືບຕໍ່.

ການຕັ້ງຄ່າສະພາບແວດລ້ອມການພັດທະນາ

8. ກົດປຸ່ມ Install ເພື່ອເລີ່ມຕົ້ນການຕິດຕັ້ງ.



ການຕັ້ງຄ່າສະພາບແວດລ້ອມການພັດທະນາ

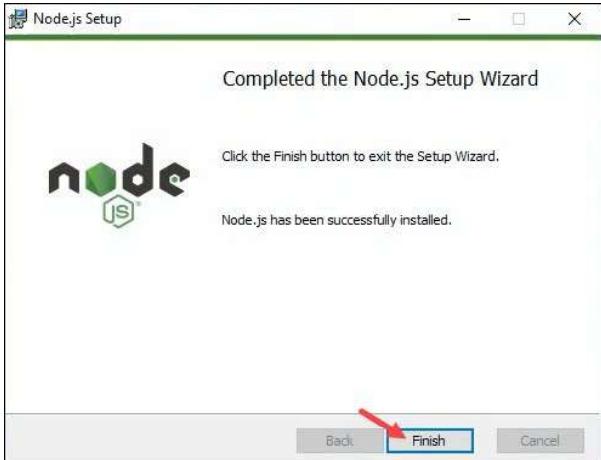
9. ຕົວຕິດຕັ້ງຮຽກຮ້ອງໃຫ້ຜູ້ເບິ່ງແຍງລະບົບຢືນຢັນເພື່ອເຮັດການປ່ຽນແປງໃສ່ອຸປະກອນ.



ໄສ່ລະຫັດຜ່ານຂອງຜູ້ດູແລລະບົບຖ້າຖືກຖາມ ແລະ ກົດ Yes ເພື່ອສືບຕໍ່.

ການຕັ້ງຄ່າສະພາບແວດລ້ອມການພັດທະນາ

10. ການຕິດຕັ້ງໃຊ້ເວລາໄລຍະໜຶ່ງ. ເມື່ອສໍາເລັດແລ້ວ, ໝໍາຈຳສຸດທ້າຍຈະສະແດງຂໍ້ຄວາມບອກວ່າຕິດຕັ້ງສໍາເລັດ.



ກົດ Finish ເພື່ອສິ້ນສຸດການຕິດຕັ້ງ ແລະ ປິດຕົວຕິດຕັ້ງ.

ການຕັ້ງຄ່າສະພາບແວດລ້ອມການພັດທະນາ

ຂັ້ນຕອນທີ 3: ກວດສອບການຕິດຕັ້ງ

ເພື່ອຢືນຢັນວ່າ Node.js ຕິດຕັ້ງສໍາເລັດແລ້ວ, ໃຫ້ໃສ່ຄໍາສັ່ງຕໍ່ໄປນີ້ໃນ command prompt ຫຼື PowerShell:

```
node -v
```

ຄໍາສັ່ງນີ້ຈະສະແດງເວັບຂັ້ນຂອງ Node.js ທີ່ຕິດຕັ້ງຢູ່ໃນລະບົບຂອງທ່ານ. ໃຫ້ໃຊ້ຄໍາສັ່ງຕໍ່ໄປນີ້ເພື່ອກວດສອບ NPM:

```
npm -v
```

```
C:\Users\KB>node -v
v20.10.0

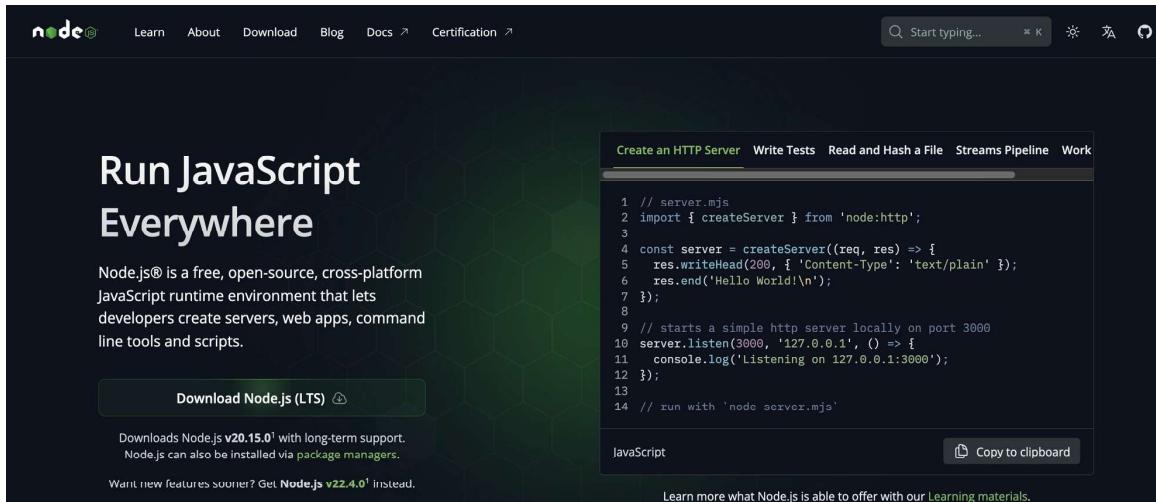
C:\Users\KB>npm -v
10.2.3

C:\Users\KB>
```

ການຕັ້ງຄ່າສະພາບແວດລ້ອມການພັດທະນາ

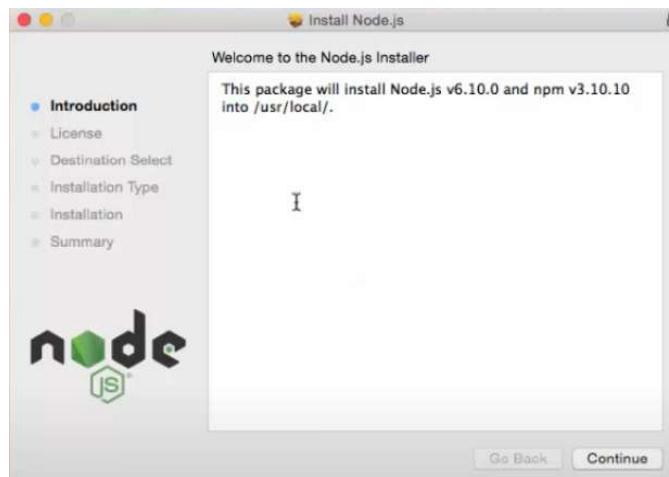
ສໍາລັບ MacOS

ຂັ້ນຕອນທີ 1: ດາວໂຫລດ Node.js ຈາກລົ້ງນີ້. ທ່ານຈະເຫັນສອງເວີຊັ້ນຄື: LTS, ເຊິ່ງຫຍໍມາຈາກ Long Term Support, ແລະເວີຊັ້ນລ່າສຸດ. ໃຫ້ດາວໂຫລດເວີຊັ້ນ LTS ເພາະວ່າມັນຖືກຮອງຮັບໃນໄລຍະຍາວ ແລະເປັນເວີຊັ້ນທີ່ໝັ້ນຄົງ.



ການຕັ້ງຄ່າສະພາບແວດລ້ອມການພັດທະນາ

ຂັ້ນຕອນທີ 2: ດຳເນີນການຕົວຕິດຕັ້ງ Node.js. ອັນນີ້ຈະເລີ່ມຕົ້ນຕົວຊ່ວຍສ້າງການຕິດຕັ້ງ, ເຮັດໃຫ້ມັນງ່າຍຕໍ່ການຕິດຕາມ. ພຽງແຕ່ຄລິກ “continue” ເມື່ອຕົວຊ່ວຍສ້າງເປີດຂຶ້ນ.



ການຕັ້ງຄ່າສະພາບແວດລ້ອມການພັດທະນາ

Step 3: ກົດໃສ່ປຸ່ມ “Agree”



ການຕັ້ງຄ່າສະພາບແວດລ້ອມການພັດທະນາ

ຂັ້ນຕອນທີ 4: ກົດໃສ່ "Install for all users of this computer".



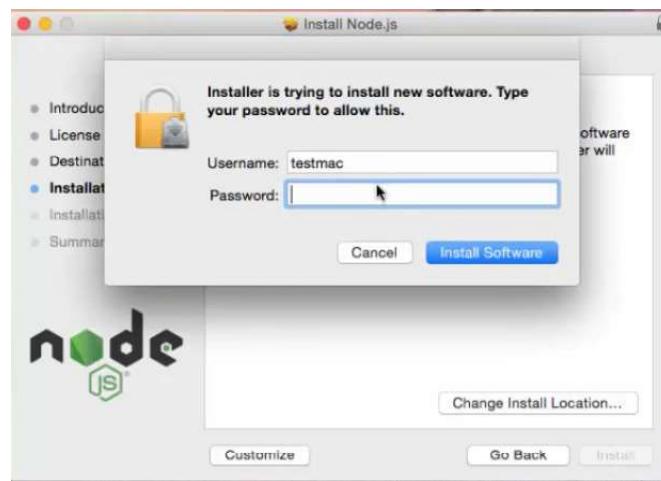
ການຕັ້ງຄ່າສະພາບແວດລ້ອມການພັດທະນາ

ຂັ້ນຕອນທີ 5: ດຽວນີ້ໃຫ້ກົດໃສ່ປຸ່ມ "install".



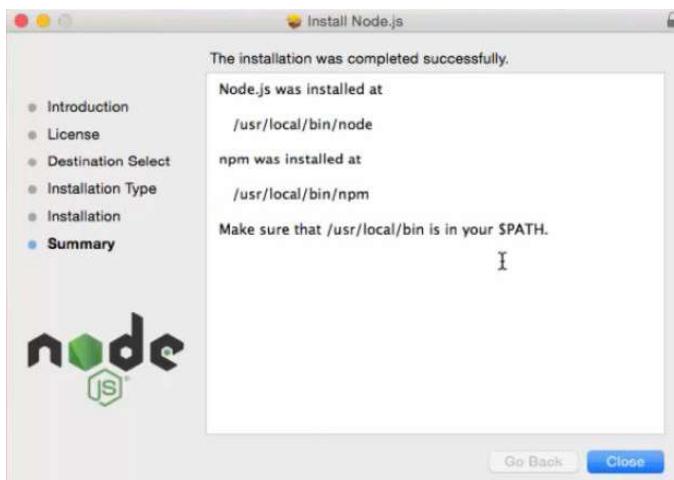
ການຕັ້ງຄ່າສະພາບແວດລ້ອມການພັດທະນາ

ຂັ້ນຕອນທີ 6: ດຽວນີ້ໃຫ້ໃສ່ລະຫັດຜ່ານລະບົບຂອງທ່ານ ແລະ ກົດໃສ່ "install software"



ການຕັ້ງຄ່າສະພາບແວດລ້ອມການພັດທະນາ

ຂັ້ນຕອນທີ 7: ຫຼັງຈາກການຕິດຕັ້ງສໍາເລັດ, ປ້ອງຢູ່ຮູມນີ້ຈະປາກົດຂຶ້ນ, ດຽວນີ້ໃຫ້ກົດໃສ່ປຸ່ມ "close".



ການຕັ້ງຄ່າສະພາບແວດລ້ອມການພັດທະນາ

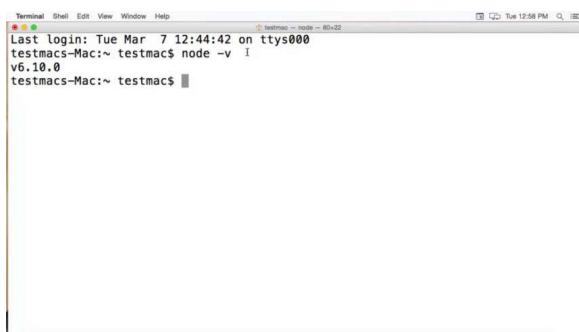
ກວດສອບການຕິດຕັ້ງ Node.js

ປະຕິບັດຕາມຂັ້ນຕອນລຸ່ມນີ້ ເພື່ອກວດສອບການຕິດຕັ້ງ Node.js:

ຂັ້ນຕອນທີ 1: ເປີດ Terminal ຂຶ້ນມາ ແລະ ພິມຄໍາສັ່ງຕໍ່ໄປນີ້

```
node -v
```

ຖ້າ Node.js ຕີກຕິດຕັ້ງຢ່າງຖືກຕ້ອງ, ທ່ານຈະເຫັນບາງສິ່ງບາງຢ່າງທີ່ຄ້າຍຄືກັນກັບຂໍ້ຄວາມນີ້ (ແຕ່ອາດຈະບໍ່ຄືກັນທຸກປະການ):



ການຕັ້ງຄ່າສະພາບແວດລ້ອມການພັດທະນາ

ຂັ້ນຕອນທີ 2: ດຽວນີ້ພວກເຮົາຈະເປີງເວີຊັ້ນຂອງ Node Package Manager ເຊິ່ງແມ່ນ NPM ໂດຍພຽງແຕ່ພິມຄໍາສັ່ງ

```
npm -v
```

ແລະມັນກໍຈະໃຫ້ເວີຊັ້ນຂອງ NPM ນຳອົກ, ເຊິ່ງໃນກໍລະນີຂອງຂໍ້ອຍແມ່ນ 3.10.10, ທ່ານຈະເຫັນບາງຢ່າງທີ່ຄ້າຍຄືກັນ (ແຕ່ອາດຈະບໍ່ຄືກັນທັງໝົດ)



```
Terminal Shell Edit View Window Help
Last login: Tue Mar  7 12:44:42 on ttys000
testmac-Mac:~ testmac$ node -v
v6.10.0
testmac-Mac:~ testmac$ npm -v
3.10.10
testmac-Mac:~ testmac$
```

ການຕັ້ງຄ່າສະພາບແວດລ້ອມການພັດທະນາ

ການຕິດຕັ້ງ Postman ໃນ Mac

- ໄປທີ່ເວັບໄຊທີ່ທາງການຂອງ Postman ທີ່ <https://www.postman.com/downloads/>
- ກົດ "Download" ສໍາລັບ Mac - ມັນຈະກວດຫາລະບົບຂອງທ່ານໂດຍອັດຕະໂນມັດ
- ໄຟລ໌ .dmg ຈະຖືກດາວໂຫລດໄປທີ່ໄຟນເດີ Downloads ຂອງທ່ານ
- ຕັບເບີນຄລິກາທີ່ໄຟລ໌ .dmg ທີ່ດາວໂຫລດມາເພື່ອ mount ມັນ
- ລາກໄອຄອນແຮ້ບ Postman ໄປໃສ່ໄຟນເດີ Applications ຂອງທ່ານ
- ເປີດໄຟນເດີ Applications ແລະດັບເບີນຄລິກ Postman ເພື່ອເປີດໃຊ້ງານມັນ
- ຖ້າທ່ານເຫັນຄໍາເຕືອນດ້ານຄວາມປອດໄພ, ໃຫ້ໄປທີ່ System Preferences > Security & Privacy ແລະກົດ "Open Anyway"

ການຕັ້ງຄ່າສະພາບແວດລ້ອມການພັດທະນາ

ການຕິດຕັ້ງ Postman ໃນ Windows

- ໄປທ໌ <https://www.postman.com/downloads/>
- ກົດ "Download" ສໍາລັບ Windows (64-bit ຫຼື 32-bit ຂຶ້ນກັບລະບົບຂອງທ່ານ)
- ໄຟລ໌ຕົວຕິດຕັ້ງ .exe ຈະຖືກດາວໂຫລດໄປທ໌ໂຟນເດີ Downloads ຂອງທ່ານ
- ຕັ້ນເບີນຄລິກາທີ່ໄຟລ໌ .exe ຂຶ້ນຈາວໂຫລດມາ
- ປະຕິບັດຕາມຂັ້ນຕອນການຕັ້ງຄ່າຂອງຕົວຊ່ວຍສ້າງການຕິດຕັ້ງ
- ເລືອກສະຖານທີ່ຕິດຕັ້ງ (ປົກກະຕິແລ້ວຕັ້ງຄ່າເລີ່ມຕົ້ນແມ່ນໃຊ້ໄດ້)
- ກົດ "Install" ແລະ ລົ້າຈົນສໍາເລັດ
- Postman ຈະເປີດຂຶ້ນໂດຍອັດຕະໂນມັດຫຼັ້ງຈາກການຕິດຕັ້ງ

ການຕັ້ງຄ່າສະພາບແວດລ້ອມການພັດທະນາ

ການຕິດຕັ້ງ Visual Studio Code ໃນ Mac

- ໄປທ໌ <https://code.visualstudio.com/>
- ກົດ "Download for Mac" (ມັນຈະກວດຫາລະບົບຂອງທ່ານໂດຍອັດຕະໂນມັດ)
- ດາວໂຫຼດໄຟລ໌ .zip ໄປທ໌ໂຟນເດີ Downloads ຂອງທ່ານ
- ຕັ້ນເບີນຄລິກາທີ່ໄຟລ໌ .zip ເພື່ອແຕກໄຟລ໌
- ລາກ "Visual Studio Code.app" ໄປທ໌ໂຟນເດີ Applications ຂອງທ່ານ
- ເປີດ Applications ແລະ ດັບເບີນຄລິກ VS Code ເພື່ອເປີດໃຊ້ງານ

ການຕັ້ງຄ່າສະພາບແວດລ້ອມການພັດທະນາ

ການຕັດຕັ້ງ Visual Studio Code ໃນ Windows

- ໄປທ໌ <https://code.visualstudio.com/>
- ກົດ "Download for Windows"
- ເລືອກລະຫວ່າງ:
 - System Installer (ແນະນຳ) - ຕິດເຕີງສໍາລັບຜູ້ໃຊ້ທຸກຄົນ
 - User Installer - ຕິດຕັ້ງສໍາລັບຜູ້ໃຊ້ປັດຈຸບັນເທົ່ານັ້ນ
- Zip - ເວີຊັ້ນແບບພິກພາ
- ດໍາເນີນການໄຟລີຕົວຕິດຕັ້ງ .exe ຫຼືດາວໂຫລດມາ
- ປະຕິບັດຕາມຕົວຊ່ວຍສ້າງການຕິດຕັ້ງ:
- ຍອມຮັບຂໍຕິກລົງໃບອະນຸຍາດ
- ເລືອກສະຖານທີ່ຕິດຕັ້ງ
- ເລືອກວຽກເພີ່ມເຕີມ (ແນະນຳໃຫ້ກວດເບື້ງ "Add to PATH")
- ກົດ "Install" ແລະລຶ້ຖ້າຈົນສໍາເລັດ

ໂປເຈັກ NodeJS ທຳອິດຂອງທ່ານ

ກວດສອບການຕິດຕັ້ງ: ເປີດ terminal/command prompt ຂອງທ່ານຂຶ້ນມາ ແລະ ພິມຄຳສິ່ງ:

```
node --version  
npm --version
```

ຂຶ້ນຕອນທີ 1: ສ້າງໄດ້ເຮັກທີ່ໂປເຈັກຂອງທ່ານ

```
mkdir my-first-node-app  
cd my-first-node-app
```

ຂຶ້ນຕອນທີ 2: ການເລີ່ມຕົ້ນໂປເຈັກ Node.js ຂອງທ່ານ

```
npm init -y
```

ສິ່ງນີ້ຈະສ້າງໄຟລ໌ package.json ດ້ວຍການຕັ້ງຄ່າເລີ່ມຕົ້ນ. ເຄື່ອງໝາຍ -y ຈະຍອມຮັບການຕັ້ງຄ່າເລີ່ມຕົ້ນທັງໝົດ.

ໂປເຈັກ NodeJS ທຳອິດຂອງທ່ານ

ຂັ້ນຕອນທີ 3: ສ້າງໄຟລ໌ Node.js ທຳອິດຂອງທ່ານ
ສ້າງໄຟລ໌ທີ່ມີຊື່ວ່າ app.js



```
// app.js
console.log("Hello World from Node.js!");

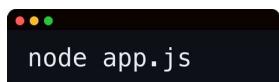
// Let's also create a simple function
function greetUser(name) {
    return `Hello, ${name}! Welcome to Node.js!`;
}

console.log(greetUser("Developer"));

// Show some Node.js specific information
console.log("Node.js version:", process.version);
console.log("Platform:", process.platform);
```

ໂປເຈັກ NodeJS ທຳອິດຂອງທ່ານ

ຂັ້ນຕອນທີ 4: ຮັນໂປຣແກຣມ Node.js ທຳອິດຂອງທ່ານ



```
node app.js
```

ທ່ານຄວນຈະເຫັນຜົນອອກມາຄ້າຍຄືກັບ:



```
Hello World from Node.js!
Hello, Developer! Welcome to Node.js!
Node.js version: v18.17.0
Platform: win32
```

ບົດເຟັກຫັດ

1. "Full Stack Development" ພາຍເຖິງຫຍັງ?

- a) ການພັດທະນາສະເພາະ frontend
- b) ການສ້າງເວັບແວ້ບພລືເຄີຊັ້ນທີ່ສິນບູນ ລວມທັງ frontend, backend ແລະຖານຂໍ້ມູນ
- c) ການພັດທະນາສະເພາະ backend
- d) ການຈັດການຖານຂໍ້ມູນສະເພາະ

2. ອັນໄດ້ປີມີທີ່ ບໍ່ແມ່ນ ເທັກໂນໂລຢີ frontend ທີ່ກ່າວເຖິງໃນໜັກສູດ?

- a) HTML5
- b) CSS3
- c) MongoDB
- d) React.js

ບົດເຟັກຫັດ

3. Node.js ແມ່ນຫຍັງ?

- a) ຖານຂໍ້ມູນ
- b) ເວັບບຮາວເຊີ
- c) runtime environment ສໍາລັບການປະມວນຜົນ JavaScript ຢູ່ນອກບຮາວເຊີ
- d) CSS framework

4. npm ຫຍ້ມາຈາກຫຍັງ?

- a) Node Package Manager
- b) New Programming Method
- c) Network Protocol Manager
- d) Node Project Module

ບົດເຟັກຫັດ

5. ຄໍາສັ່ງໃດທີ່ໃຊ້ກວດສອບວ່າ Node.js ຖືກຕິດຕັ້ງຢ່າງຖືກຕ້ອງ?

- a) npm --version
- b) node --version
- c) nodejs --check
- d) npm check

6. ໄຟລີ່ໃດທີ່ຖືກສ້າງຂຶ້ນເມື່ອທ່ານໃຊ້ຄໍາສັ່ງ npm init -y?

- a) app.js
- b) index.html
- c) package.json
- d) server.js

ບົດເຟັກຫັດ

7. ຄໍາສັ່ງໃດທີ່ໃຊ້ຮັບໄຟລ໌ Node.js ທີ່ມີຂໍ້ວ່າ app.js?

- a) npm --version
- b) node --version
- c) nodejs --check
- d) npm check

8. Node.js ເວີຊັ້ນທີ່ແນະນຳໃຫ້ດາວໂຫລດແມ່ນຫຍໍງ?

- a) ເວີຊັ້ນລ່າສຸດ
- b) ເວີຊັ້ນ LTS (Long Term Support)
- c) ເວີຊັ້ນ Beta
- d) ເວີຊັ້ນໄດ້ຮັ້ງໄດ້

ບົດເຝັກຫັດ

9. ອັນໃດຕໍ່ໄປນີ້ເປັນເທິກໂນໂລຍື backend ທີ່ກ່າວເຖິງໃນຫຼັກສູດ?

- a) HTML
- b) CSS
- c) Express.js
- d) React.js

10. ຈຸດປະສົງຂອງ V8 JavaScript engine ແມ່ນຫຍັງ?

- a) ເພື່ອສ້າງໜ້າເວັບ
- b) ເພື່ອປະມວນຜົນໂຄດ JavaScript (Node.js ຕີກສ້າງຂຶ້ນບົນພື້ນຖານນີ້)
- c) ເພື່ອຈັດການຖານຂໍ້ມູນ
- d) ເພື່ອຈັດຮູບແບບໜ້າເວັບ

**ມື້ທີ 2: Express.js API ແລະ ຫຼັກການ
REST**

ຈຸດປະສົງການຮຽນຮູ້ຂອງມື້ທີ 2

ສິ່ງທີ່ຈະໄດ້ຮຽນມີ້ນີ້:

- ຕິດຕັ້ງ ແລະ ຕັ້ງຄ່າ dependencies ທີ່ຈໍາເປັນຂອງ Node.js
- ເຂົ້າໃຈແນວຄວາມຄິດຕ້ານຄວາມປອດໄພ (XSS, CORS, Rate Limiting)
- ຕັ້ງຄ່າ development workflow ດ້ວຍ Nodemon
- ສ້າງການແຍກກ່ຽວຂ້ອງເຊັດເຈນລະຫວ່າງ routes ແລະ controllers
- ສ້າງ API ທີ່ໃຊ້ງານໄດ້ພ້ອມກັບການຈັດການຂໍ້ຜິດພາດທີ່ເໝາະສົມ

ໃນຄອນທ້າຍ: ທ່ານຈະມີ Node.js API ທີ່ປອດໄພ ແລະ ມີໂຄງສ້າງທີ່ດີພ້ອມສໍາລັບການນຳໃຊ້ຈິງ.

ພາບລວມຂອງ Dependencies ທີ່ຈໍາເປັນ

Production Dependencies:

- `express` - web framework ສໍາລັບ Node.js
- `cors` - ການແບ່ງປັນຊັບພະຍາກອນແບບຂໍາມຕິ້ນກໍາເນີດ (Cross-Origin Resource Sharing)
- `helmet` - middleware ດ້ານຄວາມປອດໄພ
- `express-rate-limit` - ການຈໍາກັດອັດຕາການຮ້ອງຂໍ (Request rate limiting)
- `morgan` - ຕົວບັນທຶກການຮ້ອງຂໍ HTTP
- `dotenv` - ຕົວແປສະພາບແວດລ້ອມ (Environment variables)

Development Dependencies:

- `nodemon` - ພິສະຕາດເຊີເວີອັດຕະໂນມັດໃນລະຫວ່າງການພັດທະນາ

Express.js Framework

Express ແມ່ນຫຍັງ?

- web framework ຂອງ Node.js ທີ່ກະທັດຮັດ ແລະ ປ່ຽນແປງໄດ້
- ຊ່ວຍໃຫ້ການສ້າງເຊີເວີ ແລະ routing ງ່າຍຂຶ້ນ
- ມີການຮອງຮັບ middleware ທີ່ແຂງແຮງ
- ເປັນມາດຕະຖານອຸດສາຫະກຳສໍາລັບ Node.js API

ການຕິດຕັ້ງ:

```
npm install express
```

ເປັນຫຍັງຕ້ອງ Express? ເຮັດໃຫ້ວຽກງານເຊີເວີທີ່ສັບສົນກາຍເປັນເລື່ອງຈ່າຍດ້ວຍໂຄດທີ່ສະອາດ ແລະ ອ່ານຈ່າຍ.

Development vs Production Dependencies

ແນວຄວາມຄິດຫຼັກ: ປະເພດຂອງ Dependencies

Production Dependencies (dependencies):

- ຈະເປັນສໍາລັບແຊັບຂອງທ່ານເພື່ອໃຫ້ເຮັດວຽກໄດ້ໃນ production
- ຄິດຕັ້ງດ້ວຍ npm install

Development Dependencies (devDependencies):

- ຕ້ອງການສະເພາະໃນລະຫວ່າງການພັດທະນາ
- ຄິດຕັ້ງດ້ວຍຄໍາສັ່ງ npm install --save-dev ຫຼື npm install -D
- ບໍ່ຖືກຕິດຕັ້ງໃນຂັ້ນຕອນ production ດ້ວຍຄໍາສັ່ງ npm install --production

ຕົວຢ່າງ: Nodemon ເປັນ devDependency - ບໍ່ຈະເປັນໃນ production.

Nodemon - პასიდთი მანქანის შემთხვევა

Nodemon მარტივია?

- რესურსების გადაცემის გარეშე მომდევნო დოკუმენტის მიზნების შემთხვევა
- უძლიერი განვითარების მიზნების შემთხვევა
- უძლიერი განვითარების მიზნების შემთხვევა

განვითარება:

```
npm install -D nodemon
```

scripts ინ Package.json

```
{
  "scripts": {
    "start": "node server.js",
    "dev": "nodemon server.js"
  }
}
```

განვითარება: npm run dev სამართლებრივი მიზნები, npm start სამართლებრივი მიზნები

განპირობებული XSS (Cross-Site Scripting)

XSS მარტივია?

- script ან სტატიკული კოდის გადაცემის გარეშე მომდევნო დოკუმენტის მიზნების შემთხვევა
- სამადლევი არა მუნაველი გარეშე მომდევნო დოკუმენტის მიზნების შემთხვევა
- მიმღები მიმღები გარეშე მომდევნო დოკუმენტის მიზნების შემთხვევა

Helmet განვითარება:

```
npm install helmet
```

- განვითარება მიმღები მიმღები გარეშე მომდევნო დოკუმენტის მიზნების შემთხვევა
- განვითარება მიმღები მიმღები გარეშე მომდევნო დოკუმენტის მიზნების შემთხვევა
- განვითარება მიმღები მიმღები გარეშე მომდევნო დოკუმენტის მიზნების შემთხვევა

კონკრეტული მიმღები გარეშე მომდევნო დოკუმენტის მიზნების შემთხვევა: კონკრეტული მიმღები გარეშე მომდევნო დოკუმენტის მიზნების შემთხვევა.

ການປ້ອງກັນດ້ວຍການຈໍາກັດອັດຕາການຮ້ອງຂໍ (Rate Limiting Protection)

ເປັນຫຍຸ້ງຕ້ອງມີ Rate Limiting?

- ປ້ອງກັນການໃຊ້ API ໃນທາງທີ່ຜິດ ແລະ ການໂຈມຕີແບບ DoS
- ຈໍາກັດຈຳນວນການຮ້ອງຂໍຕໍ່ IP address
- ປົກປ້ອງຊັບພະຍາກອນຂອງເຊື່ອເວີ

ການຈັດຕັ້ງປະຕິບັດ:

```
npm install express-rate-limit
```

ຕົວຢ່າງການຕັ້ງຄ່າ:

- 100 ຄຳຮ້ອງຂໍຕໍ່ 15 ນາທີຕໍ່ IP
- ສາມາດປັບແຕ່ງຂ່າວເວລາ ແລະ ຂີດຈຳກັດໄດ້
- ສະກັດກັ້ນການຮ້ອງຂໍທີ່ເກີນຂີດຈໍາກັດໂດຍອັດຕະໂນມັດ

ສະຖານະການຕົວຈິງ: ການປົກປ້ອງ login endpoints ຈາກການໂຈມຕີແບບ brute force

ການບັນທຶກການຮ້ອງຂໍດ້ວຍ Morgan

ເປັນຫຍຸ້ງຕ້ອງມີການບັນທຶກ (Logging)?

- ດີດຕາມການນຳໃຊ້ ແລະ ປະສິດທິພາບຂອງ API
- ແກ້ໄຂບັນຫາໃນຂັ້ນຕອນການພັດທະນາ
- ດີດຕາມຂໍ້ຜິດພາດ ແລະ ກິດຈະກຳທີ່ໜ້າສົງໄສ

ຄຸນສົມບັດຂອງ Morgan:

```
npm install morgan
```

- ຮູບແບບການບັນທຶກທີ່ຫຼາກຫຼາຍ (dev, combined, common)
- ການບັນທຶກການຮ້ອງຂໍ HTTP ໂດຍອັດຕະໂນມັດ
- ຜົນລັບທີ່ສາມາດປັບແຕ່ງໄດ້

Development vs Production: ໃຊ້ຮູບແບບ 'dev' ໃນເຄື່ອງຂອງທ່ານ, ແລະ ໃຊ້ 'combined' ໃນ production.

ການນຳໃຊ້ Environment Variables ດ້ວຍ Dotenv

ບັນຫາ:

- ລະຫັດ API keys ແລະ URL ຂອງຖານຂໍ້ມູນບໍ່ຄວນຢູ່ໃນໂຄດ
- ການຕັ້ງຄ່າທີ່ແຕກຕ່າງກັນສໍາລັບການພັດທະນາ/production
- ຄວາມສ່ຽງດໍານາຄວາມປອດໄພທີ່ຂໍ້ມູນທີ່ລະອຽດອ່ອນຖືກບັນທຶກໄວ້

ວິທີແກ້ໄຂ

```
npm install dotenv
```

ສ້າງ .env ພາຍ:

```
PORT=3000  
NODE_ENV=development  
API_KEY=your_secret_key
```

ສໍາຄັນ: ໃຫ້ເພີ່ມ .env ໄສ່ໃນ .gitignore ສະເໜີ!

Server.js - ຈຸດເຂົ້າສູ່ລະບົບແອັບພລິເຄຊັນ

ຈຸດປະສົງ:

- ເລີ່ມຕົ້ນ HTTP server
- ໂຫຼດ environment variables
- ຄວາມຮັບຜິດຊອບໜ້ອຍທີ່ສຸດ ແລະເນັ້ນທີ່ຈຸດສໍາຄັນ

ແນວຄວາມຄິດຫຼັກ:

- ນຳເຂົ້າການຕັ້ງຄ່າແອັບ Express
- ຟັງຢູ່ໃນພອດທີ່ກໍານົດໄວ້
- ຈັດການການບັນທຶກການເລີ່ມຕົ້ນຂອງເຊີເວີ

ຄວາມຮັບຜິດຊອບດຽວ: ພຽງແຕ່ເລີ່ມເຊີເວີ, ບໍ່ມີຢ່າງອື່ນ

App.js - สูมงานกานตั้งค่า Express

จุดประสงค์:

- ตั้งค่า middleware ทั่วไป
- ตั้งค่าอิทธิภาพด้านความปลอดภัย
- กำหนด global routes และกานจัดกานขั้นตอนพารา

พากส่วนสำคัญ:

- Security middleware (Helmet, CORS)
- กานอิทธิภาพ และกานบันทึกกานร่องรอย
- กานเข้าออกตัว Route
- กานจัดกานขั้นตอนพารา

กิดว่า app.js เป็นสูมควบคุมสำลับแอ็ปพลิเคชันทั้งหมดของท่าน.

Routes - กานกำหนด Endpoint ของ API

Routes มีหมายที่เรียดหยัง?

- กำหนดว่า API ของท่านจะตอบสะท้อนอย่างไร URL ใดแล้ว
- เข้าออกโดยอิทธิภาพร่องรอย HTTP (GET, POST, และอื่นๆ) เข้ากับฟังชันของ controller
- จัดกานตัวกำหนด URL และโลจิกของ routing

กานรับผิดชอบของ Route:

- กำหนดพื้นที่ endpoints
- มอบหมายอุปกรณ์ให้ controllers
- รักษาไฟล์ routes ให้คงที่และเน้นสะพานวຽກที่กว้างขึ้น

ตัวอย่าง: /api/users/:id เข้ากับ UserController.getUserById.

Controllers - Business Logic

Controllers มີຫນ້າທີ່ເຮັດຫຍຸງ?

- ປະກອບດ້ວຍໄລຈິກຂອງເອັບພລິເຄຊັ້ນຕົວຈິງ
- ປະມວນຜົນການຮ້ອງຂໍ ແລະສ້າງການຕອບສະໜອງ
- ຈັດການການກວດສອບຄວາມຖືກຕ້ອງຂອງຂໍ້ມູນແລະກໍລະນີຂໍຜິດພາດ

ຄວາມຮັບຜິດຊອບຂອງ Controller:

- ວິເຄາະຂໍ້ມູນການຮ້ອງຂໍ
- ສົ່ງຄືນການຕອບສະໜອງທີ່ມີໂຄງສ້າງ
- ຈັດການຂໍຜິດພາດຢ່າງລະວຽດ

Fat controllers, thin routes (ຄວບຄຸມການເຮັດວຽກຂອງ controllers ໃຫ້ໜັກແໜ້ນ ແລະ routes ໃຫ້ບາງເປົາ).

ເຫັກນິກການຈັດການຂໍຜິດພາດ

ການຕອບສະໜອງຂໍຜິດພາດແບບສອດຄ່ອງກັນ:

```
{  
  "success": false,  
  "message": "Error description",  
  "error": "Technical details (dev only)"  
}
```

ລະດັບການຈັດການຂໍຜິດພາດ

- ການໃຊ້ try/catch ໃນລະດັບ Controller
- Middleware ຈັດການຂໍຜິດພາດທົ່ວໄລກ
- 404 handler ສໍາລັບ routes ທີ່ບໍ່ຮັບຈັກ

Production vs Development: ເຊື່ອງລາຍລະອຽດທາງດ້ານເຫັກນິກໃນ production.

ມາດຕະຖານການຕອບສະໜອງຂອງ API

ຮູບແບບຄວາມສໍາເລັດທີ່ສອດຄ່ອງກັນ:

```
{  
  "success": true,  
  "message": "Operation completed",  
  "data": { /* actual data */ },  
  "timestamp": "2024-01-15T10:30:00Z"  
}
```

ຜົນປະໂຫຍດ:

- ໂຄງສ້າງການຕອບສະໜອງທີ່ຄາດເດີາໄດ້
- ງ່າຍຕໍ່ການເຊື່ອມໂຍງກັບ frontend
- ການບັງບອກຄວາມສໍາເລັດ/ຂໍຜິດພາດທີ່ຊັດເຈນ

ລວມມື: ລະຫັດສະຖານະ, ຕາຕະລາງເວລາ, ແລະຂໍ້ຄວາມທີ່ມີຄວາມໝາຍ

WorkFlow ການພັດທະນາ

ຂະບວນການພັດທະນາປະຈຳວັນ:

- ເລີ່ມຕົ້ນດ້ວຍຄຳສັ່ງ `npm run dev` (ໃຊ້ nodemon).
- ປັບປຸງໂຄດ ຕາມທີ່ຕ້ອງການ.
- ເຊີວີຈະຮືສະຕາດໂດຍອັດຕະໂນມັດ.
- ທົດສອບ endpoints ດ້ວຍ `curl` ຫຼື Postman.
- ກວດສອບ logs ເພື່ອຊອກຫາຂໍຜິດພາດ.
- ເຮັດຂຶ້ນຕອນຄົງກ່າວຊ້າ ໄປເລື້ອຍໆ.

ຄວາມຮັບຜິດຊອບຂອງ Controller:

- ກຳນົດຄ່າ `NODE_ENV=production`
- ໃຊ້ຄຳສັ່ງ `npm start` (ໂດຍບໍ່ມີ nodemon)
- ຕິດຕາມ logs ແລະ ປະສິດທິພາບ

ການແກ້ໄຂບັນຫາທົ່ວໄປ

ຂໍ້ຜິດພາດ "Missing parameter name"

- ກວດເບື່ງການກຳນົດຄໍາ routes ວ່າມີຂໍ້ຜິດພາດທາງໄວຍາກອນບໍ່
- ກວດໃຫ້ແນ່ໃຈວ່າໄດ້ບັນທຶກໄຟລ໌ທີ່ງໝົດຢ່າງຖືກຕ້ອງແລ້ວ
- ກວດສອບເສັ້ນທາງຂອງໄຟລ໌ໃນຄໍາສົ່ງ require

Port ກໍາລັງຖືກໃຊ້ງານຢູ່ແລ້ວ:

- ປ່ຽນ PORT ໃນໄຟລ໌ .env
- ປິດການເຮັດວຽກທີ່ກໍາລັງໃຊ້ງານຢູ່: lsof -ti:3000 | xargs kill

Module Not Found:

- ລັ້ນຄໍາສົ່ງ npm install ເພື່ອຕິດຕັ້ງ dependencies
- ກວດເບື່ງເສັ້ນທາງຂອງໄຟລ໌ ແລະ ການສະກິດຄໍາ

ອ່ານຂໍ້ຄວາມຜິດພາດຢ່າງລະມັດລະວັງຢູ່ສະເໜີ - ມັນຈະບອກທ່ານຢ່າງແນ່ນອນວ່າສົ່ງທີ່ຜິດພາດແມ່ນຫຍັງ!

ບົດເຝັກທັດ

1. Express.js ແມ່ນຫຍັງ?

- a) ຖານຂໍ້ມູນ
- (b)** web framework ທີ່ກະທັດຮັດ ແລະ ປ່ຽນແປງໄດ້ຂອງ Node.js
- c) library ຂອງ frontend
- d) ເລື່ອງມີທິດສອບ

2. ຄໍາສົ່ງໃດທີ່ໃຊ້ຕິດຕັ້ງ Express.js?

- (a)** npm install express
- b) npm get express
- c) npm add express
- d) npm download express

ບົດເຝັກຫັດ

3. ຄວາມແຕກຕ່າງກັນລະຫວ່າງ dependencies ແລະ devDependencies ແມ່ນຫຍັງ?

- a) ບໍ່ມີຄວາມແຕກຕ່າງ
- b) Dependencies ແມ່ນສໍາລັບການນຳໃຊ້ໂຈງ, ສ່ວນ devDependencies ແມ່ນສໍາລັບການພັດທະນາເທົ່ານັ້ນ
- c) devDependencies ມີຄວາມສໍາຄັນກວ່າ
- d) Dependencies ແມ່ນທາງເລືອກ

4. nodemon ເຮັດຫຍັງ?

- a) ຕິດຕາມການຈະລາຈອນຂອງເຄືອຂ່າຍ
- b) ສັງໃຫ້ເຊີບເວີເລີ່ມຕົ້ນໃໝ່ໂດຍອັດຕະໂນມັດເນື້ອມີການປ່ຽນແປງໄຟລ໌
- c) ຈັດການຖານຂໍ້ມູນ
- d) ຈັດການການປິ່ນຢັນຕົວຕົນຂອງຜູ້ໃຊ້

ບົດເຝັກຫັດ

5. XSS ຫຍ້ມາຈາກຫຍັງ?

- a) Extra Server Security
- b) Cross-Site Scripting
- c) Extended System Support
- d) External Style Sheets

6. ຂອບແເທິ່ງເປັນຄົວເຊື້ອມຕໍ່ (middleware) ໄດ້ທີ່ໃຫ້ສ່ວນຫົວ (header) ຂອງ HTTP ທີ່ກ່ຽວຂ້ອງກັບຄວາມປອດໄພ?

- a) morgan
- b) cors
- c) helmet
- d) express-rate-limit

ບົດເຝັກຫັດ

7. ຈຸດປະສົງຂອງ CORS ແມ່ນຫຍັງ?

- a) ເພື່ອຈຳກັດອັດຕາການຮ້ອງຂໍ
- **(b)** ເພື່ອເປີດໃຊ້ Cross-Origin Resource Sharing
- c) ເພື່ອ hash ລະຫັດຜ່ານ
- d) ເພື່ອບັນທຶກການຮ້ອງຂໍ

8. ໄຟລ໌ໄດ້ທີ່ຄວນບັນຈຸຂໍ້ມູນທີ່ລະອຽດອ່ອນເຊັ່ນ API keys?

- a) package.json
- b) app.js
- **(c)**.env
- d) server.js

ບົດເຝັກຫັດ

9. ໂຄງສ້າງທີ່ແມະນຳສໍາລັບການຈັດລະບຽບໂຄດແມ່ນຫຍັງ?

- a) ບຸກປ່າງຢູ່ໃນໄຟລ໌ດຽວ
- **(b)** ແຍກ routes ແລະ controllers
- c) ມີແຕ່ routes, ບໍ່ມີ controllers
- d) ການຈັດລະບຽບແບບສຸມ

10. ລະຫັດສະຖານະ HTTP ໄດ້ທີ່ບໍ່ຈະອກວ່າ "Not Found"?

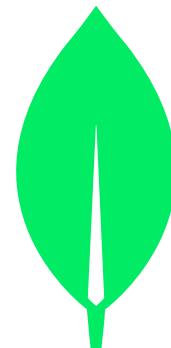
- a) 200
- b) 500
- **(c)** 404
- d) 401

ມື້ທີ 3: ການເຊື່ອມໂຍງຖານຂໍ້ມູນ - MongoDB ແລະ Mongoose

ພາບລວມຂອງ MongoDB - ຖານຂໍ້ມູນແບບ NoSQL

MongoDB ແມ່ນຫຍັງ?

- ຖານຂໍ້ມູນ NoSQL ແບບເນັ້ນເອກະສານ
- ເກັບຂໍ້ມູນເປັນເອກະສານຄໍາຍົດ JSON (BSON)
- ໂຄງສ້າງທີ່ຢືດຢູ່ນ - ບໍ່ມີໂຄງສ້າງຕາຕະລາງທີ່ແຂງຕົວ
- ສາມາດຂະໜາຍໄດ້ຕາມລວງນອນ



ເປັນຫຍັງຕ້ອງໃຊ້ MongoDB ສໍາລັບ Node.js?

- ເຂົ້າກັນໄດ້ດີກັບ JavaScript/JSON
- ພັດທະນາໄດ້ໄວດໍ່ວ່າໂຄງສ້າງທີ່ຢືດຢູ່ນ
- ສາມາດສອບຖາມໄດ້ຢ່າງຫຼາກຫຼາຍ
- ຊຸມຊົນ ແລະ ລະບົບມີເວັດທີ່ເຂັ້ມແຂງ

ພາບລວມຂອງ MongoDB - ຖານຂໍ້ມູນແບບ NoSQL

ຕົວຢ່າງ E-commerce Document:

```
{  
  "_id": "507f1f77bcf86cd799439011",  
  "name": "Wireless Headphones",  
  "price": 99.99,  
  "category": "Electronics",  
  "inStock": true,  
  "quantity": 50  
}
```

NoSQL ທຽບກັບ SQL - ຄວາມແຕກຕ່າງທີ່ສໍາຄັນ

ຖານຂໍ້ມູນແບບ SQL (MySQL, PostgreSQL):

- ໂຄງສ້າງແບບຕາຍໄຕທີ່ມີຕາຕະລາງ/ແຖວ
- ACID transactions
- ການເຊື່ອມຕໍ່ທີ່ຊັບຊ້ອນ (Complex joins)
- ການຂະຫຍາຍຕາມແນວຕັ້ງ (Vertical scaling)

ຖານຂໍ້ມູນແບບ NoSQL (MongoDB):

- ໂຄງສ້າງເອກະສານທີ່ຍືດຫຍຸນ
- ຄວາມສອດຄ່ອງໃນທີ່ສຸດ
- ເອກະສານທີ່ໂຟກຝັງແທນທີ່ຈະເປັນການເຊື່ອມຕໍ່ (joins)
- ການຂະຫຍາຍຕາມລວງນອນ (Horizontal scaling)

MongoDB Atlas - ການຕັ້ງຄ່າຖານຂໍ້ມູນຄລາວ

MongoDB Atlas ແມ່ນຫຍັງ?

- ບໍລິການ MongoDB ທີ່ໂຮດເທິງຄລາວ
- ມີຊັ້ນບໍລິການຟຣີ (ບ່ອນເກັບຂໍ້ມູນ 512MB)
- ສໍາຮອງຂໍ້ມູນ ແລະ ຄວາມປອດໄພອັດຕະໂນມັດ
- ທາງເລືອກໃນການນຳໃຊ້ທົ່ວໂລກ
- ບໍ່ຈໍາເປັນຕ້ອງຕິດຕັ້ງ

ເວັບໄຊທາງການ: <https://www.mongodb.com/atlas>

ຂັ້ນຕອນການສ້າງບັນຊີAtlas

ຂັ້ນຕອນທີ 1: ສ້າງບັນຊີ MongoDB

- ໄປທ໌ [mongodb.com/atlas](https://www.mongodb.com/atlas)
- ກົດ "Try Free"
- ໃສ່ອີເມວ, ລະຫັດຜ່ານ, ແລະຂໍ້ມູນພື້ນຖານ
- ປຶກຂໍ້ມູນທີ່ຢູ່ອີເມວ
- ສໍາເລັດການຕັ້ງຄ່າບັນຊີ

ຂັ້ນຕອນທີ 2: ການຕັ້ງຄ່າອົງກອນ ແລະ ໂຄງການ

- ເລືອກຊື່ອົງກອນ (ຕົວຢ່າງ: "My E-commerce Project")
- ສ້າງໂຄງການທຳອິດ (ຕົວຢ່າງ: "Online Store API")
- ໂຄງການຊ່ວຍຈົດລະບຽບແຮບພລິເຄຊັນທູາຍງອັນ

ສໍາຄັນ: ໃຫ້ຕິດຕາມຂໍ້ມູນເຂົ້າສູ່ລະບົບຂອງທ່ານໄວ້ໃຫ້ດີ!

ການສ້າງ Cluster ທຳອິດຂອງທ່ານ

ຂັ້ນຕອນທີ 3: ສ້າງ Cluster ທຳອິດຂອງທ່ານ

- ລົງທຶນ "Build a Database" (ສ້າງຖານຂໍ້ມູນ)
- ເລືອກຕົວເລືອກ Shared (Free) (ໃຊ້ຮ່ວມກັນ (ຟຣີ))
- ເລືອກຜູ້ໃຫ້ບໍລິການ AWS
- ເລືອກພາກພື້ນທີ່ໄກ້ກັບທ່ານທີ່ສຸດ
- ຕັ້ງຊັ້ນລັບເຄືອງທ່ານ: "ecommerce-cluster"
- ລົງທຶນ "Create Cluster" (ສ້າງຄລັບເຄືອງ)

ເວລາໃນການສ້າງຄລັບເຄືອງ:

- ໃຊ້ເວລາ 1-3 ນາທີເພື່ອຈັດສັນ
- ທ່ານຈະເຫັນໜ້າຈຳກຳລັງໂທູດ
- ຢ່າໂທູດໜ້າຈຳຄົນໃໝ່ໃນຂະນະກຳລັງຕັ້ງຄ່າ

ຂດຈາກກັດຂອງຊັ້ນບໍລິການຟຣີ: ບ່ອນເກັບຂໍ້ມູນ 512MB, CPU ແບບໃຊ້ຮ່ວມກັນ, ເພົະສີມທີ່ສຸດສໍາລັບການຮຽນຮູ້.

ການຕັ້ງຄ່າຄວາມປອດໄພຂອງຖານຂໍ້ມູນ

ຂັ້ນຕອນທີ 4: ສ້າງຜູ້ໃຊ້ຖານຂໍ້ມູນ

- ລົງທຶນ "Database Access" (ເຂົ້າເຖິງຖານຂໍ້ມູນ) ຢູ່ແຖບດ້ານຊ້າຍ
- ລົງທຶນ "Add New Database User" (ເພີ່ມຜູ້ໃຊ້ຖານຂໍ້ມູນໃໝ່)
- ເລືອກການຢືນຢັນຕົວຕົນແບບ "Password" (ລະຫັດຜ່ານ)
- ຊື່ຜູ້ໃຊ້: `ecommerce_admin`
- ລະຫັດຜ່ານ: ສ້າງລະຫັດຜ່ານທີ່ປອດໄພ (ບັນທຶກໄວ້!)
- ສິດທິພິເສດຂອງຜູ້ໃຊ້ຖານຂໍ້ມູນ: "Read and write to any database" (ອ່ານ ແລະ ຂຽນໃສ່ຖານຂໍ້ມູນໄດ້ກຳໄດ້)
- ລົງທຶນ "Add User" (ເພີ່ມຜູ້ໃຊ້)

ຂັ້ນຕອນທີ 5: ການຕັ້ງຄ່າການເຂົ້າເຖິງເດືອຂ່າຍ

- ລົງທຶນ "Network Access" (ການເຂົ້າເຖິງເດືອຂ່າຍ) ຢູ່ແຖບດ້ານຊ້າຍ
- ລົງທຶນ "Add IP Address" (ເພີ່ມທີ່ຢູ່ IP)
- ສໍາລັບການພັດທະນາ: ລົງທຶນ "Allow Access from Anywhere" (ອະນຸຍາດການເຂົ້າເຖິງຈາກທຸກບ່ອນ) (0.0.0.0/0)
- ສໍາລັບການນຳໃຊ້ຕົວຈິງ: ໃຫ້ເພີ່ມທີ່ຢູ່ IP ທີ່ເຈົ້າຈິງ
- ລົງທຶນ "Confirm" (ຢືນຢັນ)

ໝາຍເຫດດ້ານຄວາມປອດໄພ: ໃນການນຳໃຊ້ຕົວຈິງ, ຕ້ອງຈຳກັດການເຂົ້າເຖິງ IP ສະເໜີ!

ການຊອກຫາທີ່ຢູ່ IP

Windows

- ກົດປຸ່ມ Win + R, ພິມ cmd, ແລ້ວກົດ Enter
- ພິມ ipconfig ແລະ ກົດ Enter
- ຊອກຫາອະແດບເຕືອຂ່າຍຂອງທ່ານ - IP ພາຍໃນຂອງທ່ານຈະຖືກລະບຸເປັນ "IPv4 Address"
- ສໍາລັບຂໍ້ມູນລະຽດເພີ່ມເຕີມ, ໃຫ້ໃຊ້ ipconfig /all

MacOS

- ເປີດ Terminal (Applications > Utilities > Terminal)
- ພິມ ifconfig ແລ້ວກົດ Enter
- ຊອກຫາອິນເຕີເຟດເຄືອຂ່າຍຂອງທ່ານ (ໂດຍທົ່ວໄປແມ່ນ en0 ສໍາລັບ Ethernet ຫຼື en1 ສໍາລັບ Wi-Fi)
- ທີ່ຢູ່ IP ຂອງທ່ານຈະຖືກລະບຸວ່າ "inet"

ການສ້າງຖານຂໍ້ມູນ ແລະ ການເອົາ Connection String

ຂັ້ນຕອນທີ 6: ສ້າງຖານຂໍ້ມູນ

- ກັບຄືນໄປທີ່ "Database" (ຫຼື "Clusters")
- ຄລິກ "Browse Collections" (ຊອກຫາ Collections) ຢູ່ຄລິສເຕີຂອງທ່ານ
- ຄລິກ "Add My Own Data" (ເພີ່ມຂໍ້ມູນຂອງຂ້ອຍເອງ)
- ຊື່ຖານຂໍ້ມູນ: ecommerce_store
- ຊື່Collection: products
- ຄລິກ "Create" (ສ້າງ)

ຂັ້ນຕອນທີ 7: ເອົາ Connection String

- ຄລິກປຸ່ມ "Connect" (ເຊື່ອມຕໍ່) ຢູ່ພາບລວມຂອງຄລິສເຕີ
- ເລືອກ "Connect your application" (ເຊື່ອມຕໍ່ແອັບພລິເຄຊັນຂອງທ່ານ)
- ເລືອກໄດ້ຮັບເວີ "Node.js"
- ສໍາເນົາ Connection String
- ປ່ຽນແທນ <password> ດ້ວຍລະຫັດຜ່ານຜູ້ໃຊ້ຖານຂໍ້ມູນຂອງທ່ານ
- ປ່ຽນແທນ <database> ດ້ວຍ ecommerce_store

String ເຊື່ອມຕໍ່ສຸດທ້າຍ: `mongodb+srv://ecommerce_admin:yourpassword@ecommerce-cluster.xyz.mongodb.net/ecomerce_store?retryWrites=true&w=majority`

ບົດນຳ Mongoose ODM

Mongoose ແມ່ນຫຍັງ?

1. Object Document Mapper (ODM) ສໍາລັບ MongoDB
2. ໃຫ້ການກວດສອບໂຄງຮ່າງ (schema) ສໍາລັບ MongoDB
3. ມີການປ່ຽນຂະໜາດຂໍ້ມູນ (type casting) ແລະ ການກວດສອບຄວາມຖືກຕ້ອງໃນຕົວ
4. ຮອງຮັບ middleware (pre/post hooks)
5. ການສ້າງການສອບຖາມ (query) ແລະ ການນຳໃຊ້ຂໍ້ມູນ (population)

ການຕິດຕັ້ງ:

```
npm install mongoose
```

ການເຊື່ອມຕໍ່ກັບ MongoDB ດ້ວຍ Mongoose

ການຕັ້ງຄ່າການເຊື່ອມຕໍ່ພື້ນຖານ:

```
const mongoose = require('mongoose');

const connectDB = async () => {
  try {
    await mongoose.connect(process.env.MONGODB_URI);
    console.log('✅ MongoDB Connected to E-commerce Store');
  } catch (error) {
    console.error('❌ MongoDB connection failed:', error);
    process.exit(1);
  }
};
```

Environment Variables (.env):

MONGODB_URI=mongodb+srv://ecommerce_admin:yourpassword@ecommerce-cluster.xyz.mongodb.net/ecomerce_store

ການທຶດສອບການເຊື່ອມຕໍ່Atlas ຂອງທ່ານ

ກວດສອບວ່າການເຊື່ອມຕໍ່ໄຊ້ງານໄດ້:



```
// test-connection.js
require('dotenv').config();
const mongoose = require('mongoose');

const testConnection = async () => {
  try {
    await mongoose.connect(process.env.MONGODB_URI);
    console.log('✅ Successfully connected to E-commerce Database!');

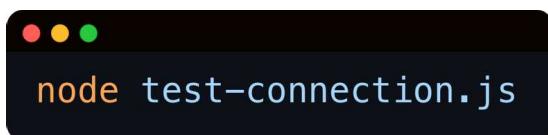
    // Test basic operation
    const collections = await mongoose.connection.db.listCollections().toArray();
    console.log('🗄 Available collections:', collections.map(c => c.name));

    await mongoose.connection.close();
    console.log('👋 Connection closed');
  } catch (error) {
    console.error('❌ Connection failed:', error.message);
  }
};

testConnection();
```

ການທຶດສອບການເຊື່ອມຕໍ່Atlas ຂອງທ່ານ

ສ້າງທຶດສອບ:



```
node test-connection.js
```

ບັນຫາທີ່ພົບເລື້ອຍ ແລະ ວິທີແກ້ໄຂ:

- ການຢືນຢັນຄົວຕົນລົ້ມໜູວ: ກວດເບິ່ງຊື່ຜູ້ໃຊ້/ລະຫັດຜ່ານ
- ໝົດເວລາຂອງເຄືອຂ່າຍ: ກວດເບິ່ງ IP whitelist
- ບໍ່ພົບຖານຂໍ້ມູນ: ຢືນຢັນຊື່ຖານຂໍ້ມູນໃນ connection string

ບົດເຝັກຫັດ

1. MongoDB ແມ່ນຖານຂໍ້ມູນປະເພດໃດ?

- a) SQL database
- b) NoSQL document-oriented database
- c) Graph database
- d) Key-value database

2. ຮູບແບບໃດທີ່ MongoDB ໃຊ້ໃນການເກັບຂໍ້ມູນ?

- a) ຕາຕະລາງ ແລະ ແຖວ
- b) ເອກະສານຄ້າຍື່ງ (BSON)
- c) ໄຟລ് XML
- d) ຮູບແບບ CSV

ບົດເຝັກຫັດ

3. MongoDB Atlas ແມ່ນຫຍັງ?

- a) local database
- b) ບໍລິການ MongoDB ແບຄລາວ (cloud-hosted MongoDB service)
- c) ໄດຣເວີຖານຂໍ້ມູນ (database driver)
- d) query language

4. ODM ທີ່ມາຈາກຫຍັງໃນ Mongoose?

- a) Object Data Model
- b) Object Document Mapper
- c) Online Database Manager
- d) Organized Data Method

ບົດເຝັກຫັດ

5. ຄວາມຈຸແບບຝຶຂອງ MongoDB Atlas ແມ່ນເທົ່າໄດ?

- a) 1GB
- b) 512MB
- c) 256MB
- d) 2GB

6. ຄໍາສົ່ງໃດທີ່ໃຊ້ໃນການຕິດຕັ້ງ Mongoose?

- a) npm install mongodb
- b) npm install mongoose
- c) npm install atlas
- d) npm install mongo

ບົດເຝັກຫັດ

7. ຄວນເຮັດແວວໃດກັບ MongoDB connection string?

- a) ໃສ່ໄວ້ໃນໂຄດຂອງທ່ານໂດຍກົງ
- b) ເກັບໄວ້ໃນ (environment variables)
- c) ແບ່ງປັນຕິສາທາລະນະ
- d) ໃສ່ໄວ້ໃນไฟລ໌ package.json

8. ຂໍ້ດີຫຼັກຂອງຖານຂໍ້ມູນ NoSQL ເມື່ອທຽບກັບ SQL ແມ່ນຫຍັງ?

- a) ໄວກວ່າສະເໜີ
- b) ໂຄງສ້າງທີ່ຢືດຫຍຸ້ນ
- c) ຄວາມປອດໄພດີກວ່າ
- d) ຂະໜາດໄຟລ໌ນ້ອຍກວ່າ

ບົດເຝັກຫັດ

9. ວິທີໄດ້ໃຊ້ໃນການເຊື້ອມຕໍ່ MongoDB ດ້ວຍ Mongoose?

- a) mongoose.connect()
- b) mongoose.start()
- c) mongoose.open()
- d) mongoose.link()

10. ເຮົາຕ້ອງເລີ້າຫຍ່ງມາແທນທີ່ <password> ໃນ MongoDB connection string?

- a) ກໍາວ່າ "password"
- b) ລະຫັດຜ່ານຕົວລົງຂອງຜູ້ໃຊ້ຖານຂໍ້ມູນຂອງທ່ານ
- c) ລະຫັດຜ່ານບັນຊີ MongoDB ຂອງທ່ານ
- d) ຂໍ້ຄວາມແບບສຸມໄດ້ກຳໄດ້

ມື້ທີ 4: ການຈັດການຜູ້ໃຊ້ ແລະ ການຢືນຢັນຕົວຕົນ (User Management & Authentication)

Authentication vs Authorization

ການຢືນຢັນຕົວຕົນ (Authentication): "ເຈົ້າແມ່ນໃຜ?"

- ການຢືນຢັນຕົວຕົນຂອງຜູ້ໃຊ້
- ການເຂົ້າສູ່ລະບົບດ້ວຍອີເມວ/ລະຫັດຜ່ານ
- ການກວດສອບຂໍ້ມູນປະຈຳຕົວທີ່ຖືກຕ້ອງ
- ການອອກໂທເຄີນ (token) ສໍາລັບຜູ້ໃຊ້ທີ່ໄດ້ຮັບການຢືນຢັນ

ການອະນຸຍາດ (Authorization): "ເຈົ້າສາມາດເຮັດຫຍັງໄດ້?"

- ການກຳນົດສິດທີ່ຂອງຜູ້ໃຊ້
- ການເຂົ້າເຖິງຕາມບົດບາດ (ຜູ້ດຸແລລະບົບ, ລູກຄ້າ)
- ຄວາມເປັນເຈົ້າຂອງຂັບພະຍາກອນ (ສະເພາະຄໍາສົ່ງຊື້ຂອງທ່ານ)
- ການຄວບຄຸມການເຂົ້າເຖິງຄຸນສົມບັດ

ຕົວຢ່າງຈາກ E-commerce ຕົວຈິງ:

- ການຢືນຢັນຕົວຕົນ (Authentication): ລູກຄ້າເຂົ້າສູ່ລະບົບເພື່ອຊື້ສິນຄ້າ
- ການອະນຸຍາດ (Authorization): ສະເພາະຜູ້ດຸແລລະບົບເທົ່ານັ້ນທີ່ສາມາດເພີ່ມສິນຄ້າໃໝ່ໄດ້, ສ່ວນລູກຄ້າສາມາດເບິ່ງໄດ້ ສະເພາະຄໍາສົ່ງຊື້ຂອງຕົນເອງ

ພາບລວມຂອງ JWT (JSON Web Tokens)

JWT ແມ່ນຫຍັງ?

- ໂທຕັນທີ່ບັນຈຸຂໍ້ມູນດ້ວຍຕົວເອງ (self-contained) ສໍາລັບການສົ່ງຂໍ້ມູນຢ່າງປອດໄພ
- ການຢືນຢັນຕົວຕົນທີ່ບໍ່ມີສະຖານະ (stateless authentication) (ບໍ່ມີເຊັ່ນຝຶ່ງເຊີບເວີ)
- ເປັນມາດຕະຖານອຸດສາຫະກໍາສໍາລັບເວັບ API
- ບັນຈຸຂໍ້ມູນຜູ້ໃຊ້ ແລະ ເວລາໝົດອາຍຸ

ໂຄງສ້າງຂອງ JWT:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c

ສາມພາກສ່ວນ:

- ສ່ວນຫົວ (Header): Algorithm ແລະ ປະເພດໂທເຄີນ
- ສ່ວນຂໍ້ມູນ (Payload): ຂໍ້ມູນຜູ້ໃຊ້ ແລະ ການອ້າງສິດ (claims)
- ລາຍເຊັນ (Signature): ລາຍເຊັນຢືນຢັນ

ການ Hash ແລະ ຫັດຜ່ານດ້ວຍ Bcrypt

ເປັນຫຍັງຕ້ອງ Hash ແລະ ຫັດຜ່ານ?

- ບໍ່ເກີບລະຫັດຜ່ານແບບຂໍ້ຄວາມທຳມະດາ
- ປ້ອງກັນຖ້າຖານຂໍ້ມູນຖືກໂຈມຕີ
- ການເຂົ້າລະຫັດແບບທາງດຽວ (ບໍ່ສາມາດປິ້ນກັບຄືນໄດ້)
- Salt ຊ່ວຍປ້ອງກັນການໂຈມຕີແບບ rainbow table

ຕິດຕັ້ງ Bcrypt:

```
npm install bcryptjs
```

ການ Hash ແລະ ຫັດຜ່ານດ້ວຍ Bcrypt

ການປະຕິບັດການ Hash ແລະ ຫັດຜ່ານ:

```
const bcrypt = require('bcryptjs');

// Hash password before saving
userSchema.pre('save', async function(next) {
  // Only hash if password is modified
  if (!this.isModified('password')) return next();

  try {
    // Generate salt and hash password
    const salt = await bcrypt.genSalt(12);
    this.password = await bcrypt.hash(this.password, salt);
    next();
  } catch(error) {
    next(error);
  }
});

// Instance method to compare passwords
userSchema.methods.comparePassword = async function(candidatePassword) {
  return await bcrypt.compare(candidatePassword, this.password);
};

// Static method to find user and check password
userSchema.statics.findByCredentials = async function(email, password) {
  const user = await this.findOne({ email }).select('+password');
  if (!user) {
    throw new Error('Invalid email or password');
  }

  const isMatch = await user.comparePassword(password);
  if (!isMatch) {
    throw new Error('Invalid email or password');
  }

  return user;
};
```

ການສ້າງ ແລະ ການປິ່ນຢັນ JWT Token

ວິທີການໃຊ້ງານ JWT Token:

```
npm install jsonwebtoken
```

Environment Variables (.env):

```
JWT_SECRET=your_super_secret_jwt_key_here_make_it_long_and_complex
JWT_EXPIRE=7d
```

JWT Token Methods:

```
const jwt = require('jsonwebtoken');

// Instance method to generate JWT token
userSchema.methods.generateAuthToken = function() {
  const payload = {
    id: this._id,
    email: this.email,
    role: this.role,
    name: this.name
  };

  return jwt.sign(
    payload,
    process.env.JWT_SECRET,
    { expiresIn: process.env.JWT_EXPIRE || '7d' }
  );
};

// Static method to verify token
userSchema.statics.verifyToken = function(token) {
  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    return decoded;
  } catch (error) {
    throw new Error('Invalid token');
  }
};

// Method to get public profile (no sensitive data)
userSchema.methods.getPublicProfile = function() {
  return {
    id: this._id,
    name: this.name,
    email: this.email,
    role: this.role,
    profile: this.profile,
    isActive: this.isActive,
    createdAt: this.createdAt
  };
};
```

ບົດເຝັກຫັດ

1. authentication ແລະ authorization ຕ່າງກັນຄືແນວໃດ?

- a) ມັນແມ່ນສິ່ງດຽວກັນ
- b) Authentication ແມ່ນ "ເຈົ້າແມ່ນໃຜ?", ສ່ວນ Authorization ແມ່ນ "ເຈົ້າສາມາດຮັດຫຍັງໄດ້?"
- c) Authorization ເກີດຂຶ້ນກ່ອນ authentication
- d) Authentication ເປັນທາງເລືອກ, ສ່ວນ authorization ແມ່ນຈຳເປັນ

2. JWT ຫຍໍ້ມາຈາກຫຍັງ?

- a) JavaScript Web Token
- b) JSON Web Token
- c) Java Web Technology
- d) Just Web Token

ບົດເຝັກຫັດ

3. JWT ມີຈັກພາກສ່ວນ?

- a) 2
- b) 3
- c) 4
- d) 5

4. ເປັນຫຍຸງລະຫັດຜ່ານຈຶ່ງຕ້ອງຖືກ hashed?

- a) ເພື່ອເຮັດໃຫ້ມັນຍາວຂຶ້ນ
- b) ເພື່ອປັກປ້ອງພວກມັນທ້າທາກຖານຂໍ້ມູນຖືກເຈາະລະບົບ
- c) ເພື່ອເຮັດໃຫ້ມັນປະມວນຜົນໄວຂຶ້ນ
- d) ເພື່ອຫຼຸດພື້ນທີ່ໃນການລັດເກັບ

ບົດເຝັກຫັດ

5. library ໄດທີໃສ່ໃນການ hash ລະຫັດຜ່ານໃນ Node.js?

- a) crypto
- b) bcrypt
- c) hash
- d) secure

6. salt ແມ່ນຫຍຸງໃນການ hashing ລະຫັດຜ່ານ?

- a) ເຕືອງປຸງອາຫານ
- b) ຂໍມູນແບບສຸມທີ່ຖືກເພີ່ມເຂົ້າເພື່ອປ້ອງກັນການໂຄມຕີແບບ rainbow table
- c) ປະເພດຂອງການເຂົ້າລະຫັດ
- d) ຊອງຂໍ້ມູນໃນທາງຂໍ້ມູນ

ບົດເຝັກຫັດ

7. JWT secrets ຄວນເກັບຢູ່ໃສ?

- a) ໃນລະຫັດ
- b) ໃນຕົວແປສະພາບແວດລ້ອມ (environment variables)
- c) ໃນຖານຂໍ້ມູນ
- d) ໃນໄຟລ໌ package.json

8. ຖ້າ JWT token ໝຶດອາຍຸຈະເກີດຫຍັງຂຶ້ນ?

- a) ມັນຈະຕ່ອງໄດຍອັດຕະໂນມັດ
- b) ຜູ້ໃຊ້ຕ້ອງເຂົ້າສູ່ລະບົບໃໝ່ອີກຄັ້ງ
- c) ມັນມີຄວາມປອດໄພຫຼາຍຂຶ້ນ
- d) ບໍ່ມີຫຍັງເກີດຂຶ້ນ

ບົດເຝັກຫັດ

9. JWT tokens ມັກຈະຖືກສົ່ງໄປນຳ HTTP header ໂຕໄດ?

- a) Content-Type
- b) Authorization
- c) Accept
- d) User-Agent

10. ອີທີທີ່ແນະນຳໃນການຢືນຢັນລະຫັດຜ່ານແມ່ນໄຕໄດ?

- a) ຫຽບເທົ່າລະຫັດຜ່ານແບບຂໍ້ຄວາມທີ່ມະດາ
- b) ໃຊ້ວິທີ bcrypt.compare()
- c) ປອດລະຫັດ hash ແລ້ວຫຽບເທົ່າ
- d) ໃຊ້ເວົ້າປະເທົດການ ==

มีติ 5: งานพัฒนา Frontend - Interface และ Styling

แนะนำว่ากับงานพัฒนาเว็บ และ React.js

งานพัฒนาเว็บใช้แม่นยัง?

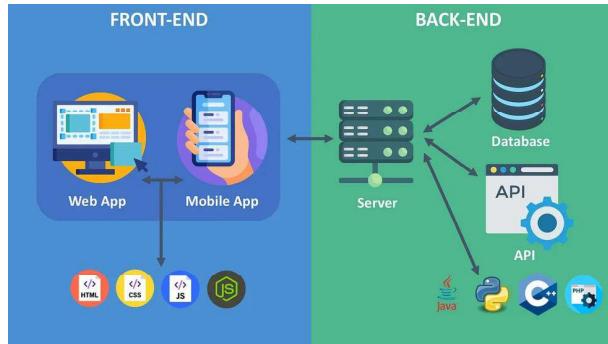
ถ้าจะทิบายกับงานพัฒนาเว็บ และ ความสำคัญของมัน



งานพัฒนาเว็บ (Web development) ประกอบมีงานสร้างเว็บไซต์ และ แอปพลิเคชันเว็บ, เข้ากับข้อกับงานออกแบบเว็บ, งานพัฒนาเนื้อหา, scripting ฝั่ง client/server, และ ความปลอดภัยของเครือข่าย. ความสำคัญของมันแม่นยำในเรื่องการเข้ารหัสแบบ dynamical และ โต้ตอบได้ (interactive) สำลับกับสิ่งที่ต้องการ, ภาษาถ้า, และ ภาษาแบบปั๊บปั๊บ.

ແນະນຳກ່ຽວກັບການພັດທະນາເວັບ ແລະ React.js

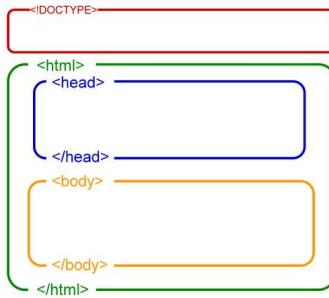
ຄວາມແຕກຕ່າງລະຫວ່າງການພັດທະນາ frontend ແລະ backend :



- Frontend Development: ເຈະຈຶ່ງໃສ່ ຜູ້ໃຊ້ງານເວັບແອັບພື້ນເຄຊັນເປັນຫຼັກ. ໂດຍສະເພາະການອອກແບບ ແລະ ການສ້າງອົງປະກອບຕ່າງໆຂອງໜ້າເວັບ ເພື່ອໃຫ້ເບິ່ງງາມ ສະບາຍຕາ ຕໍ່ຜູ້ໃຊ້ງານ, ການນຳໃຊ້ເຕັກໂນໂລຊີ ມາພັດທະນາ ແຊ່ນ: HTML, CSS, ແລະ JavaScript.
- Backend Development: ເຈະຈຶ່ງໃສ່ ຜູ້ໃຫ້ບໍລິການເວັບແອັບພື້ນເຄຊັນເປັນຫຼັກ. ໂດຍສະເພາະ ການຄຸ້ມຄອງເຄື່ອງແມ່ ຂ່າຍ, ຖານຂໍ້ມູນ, ແລະ ຕັກກະຂອງແອັບພື້ນເຄຊັນ, ການນຳໃຊ້ເຕັກໂນໂລຊີ ມາພັດທະນາ ແຊ່ນ: Node.js, Python, Ruby, Java, ແລະ ທານຂໍ້ມູນເຊັ່ນ MySQL, PostgreSQL, ຫຼື MongoDB.

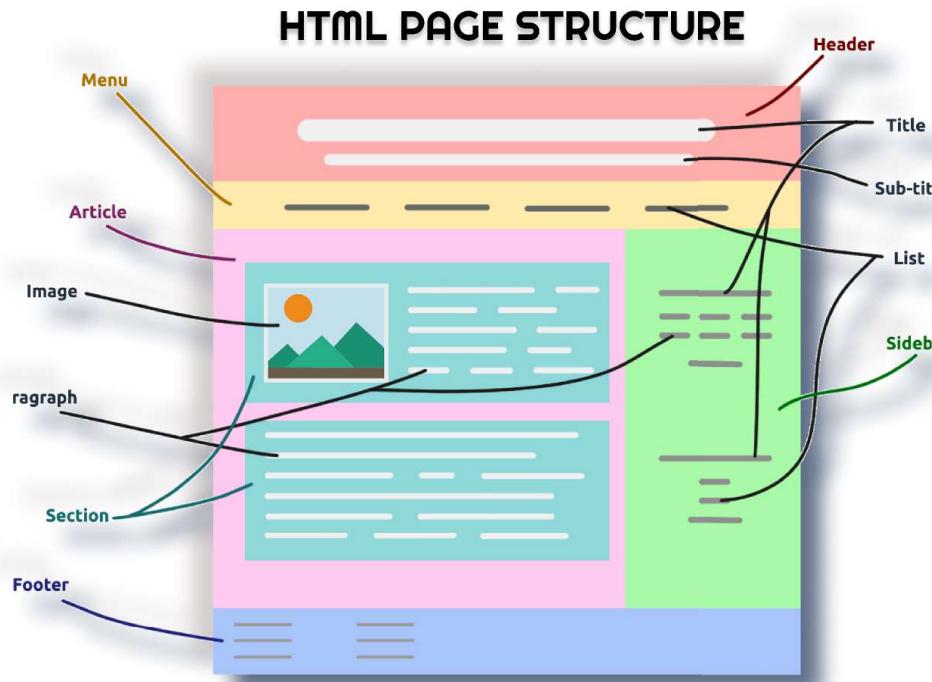
ແນະນຳກ່ຽວກັບການພັດທະນາເວັບ ແລະ React.js

ເຕັກໂນໂລຊີພື້ນຖານ
HTML (HyperText Markup Language)



- ໃນການຂຽນພາສາ HTML ເລີ່ມຕົ້ນດ້ວຍການປະກາດແທັກ <!DOCTYPE html>, ຕາມດ້ວຍແທັກ <html> ທີ່ກວມເອົາທັງສ່ວນ <head> ແລະ <body>.
- ພາກສ່ວນ <head> ມີຂໍ້ມູນເມຕາ, ລຶ້ງໄປຫາສະໄຕລົຊີສ, ແລະ ສະຄຣິບ.
- ພາກສ່ວນ <body> ມີເນື້ອຫາທີ່ສະແດງໃຫ້ຜູ້ໃຊ້, ລວມທັງຂໍ້ຄວາມ, ຮູບພາບ, ລົງ, ແລະ ອົງປະກອບເຊັ່ນງ.

ແນະນຳກ່ຽວກັບການພັດທະນາເວັບ ແລະ React.js



ແນະນຳກ່ຽວກັບການພັດທະນາເວັບ ແລະ React.js

ແຫ້ງHTML ທົ່ວໄປ (headings, paragraphs, links, images, lists, forms):

- ການສ້າງຫົວຂໍ ໃຊ້ແຫ້ງ : `<h1>` to `<h6>`.
- ການສ້າງຂໍຄວາມເປັນພາລາກຮາຟ ໃຊ້ແຫ້ງ: `<p>`.
- ການສ້າງລັ້ງ ໃຊ້ແຫ້ງ: `<a>`.
- ການສ້າງຝາຍຮູບພາບ ໃຊ້ແຫ້ງ: ``.
- ການສ້າງລາຍການ ໃຊ້ແຫ້ງ: `` for unordered lists, `` for ordered lists, `` for list items.
- ການສ້າງຟອມ ໃຊ້ແຫ້ງ: `<form>`, `<input>`, `<textarea>`, `<button>`, `<select>`, and `<option>`.

ແນະນຳກ່ຽວກັບການພັດທະນາເວັບ ແລະ React.js

CSS (Cascading Style Sheets)



CSS ຊ່ວຍໃຫ້ເວົາສາມາດນຳໃຊ້ຮູບແບບຕ່າງໆກັບອົງປະກອບ HTML ເພື່ອເພີ່ມຮູບລັກສະນະຂອງຫຼັກເວັບ. ມັນສາມາດຄວບຄຸມຮູບແບບ, ສີ, ຕົວອັກສອນ, ແລະ ການອອກແບບໂດຍລວມ.

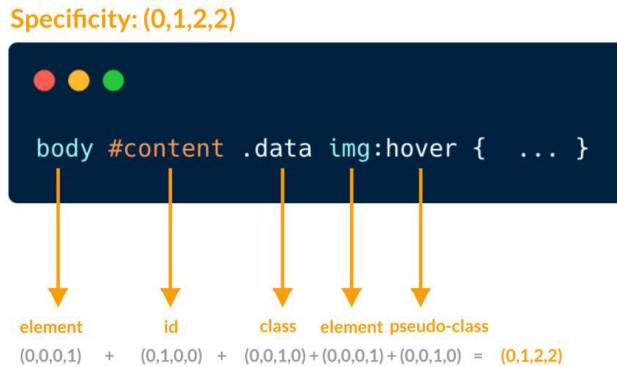
ແນະນຳກ່ຽວກັບການພັດທະນາເວັບ ແລະ React.js

ຄຸນສິນບັດCSS ພື້ນຖານ(color, background, font, border, margin, padding):

- ສີ: color, background-color.
- ພາບພື້ນຫຼັງ: background, background-image.
- ຕົວອັກສອນ: font-family, font-size, font-weight.
- ເສັ້ນ: border, border-radius.
- ຂອບ: margin.
- ການຂະຫຍາຍ: padding.

ແນະນຳກ່ຽວກັບການພັດທະນາເວັບ ແລະ React.js

ແນະນຳ ຕົວເລືອກຂອງພາສາ CSS ແລະ ຄໍາສະເພາະເຈາະຈຶງ



- ຕົວເລືອກ: ອົງປະກອບຂອງ HTML ທີ່ຕ້ອງການໃຊ້ຮູບແບບ. ເຊັ່ນຕົວຢ່າງ ຕົວເລືອກອົງປະກອບ (`h1`, `p`), ຕົວເລືອກຄລາສ (`.classname`), ແລະ ຕົວເລືອກ ID (`#idname`).
- ຄໍາສະເພາະ: ກໍານົດວ່າຈະເລືອກໃຊ້ຮູບແບບໃດ ເມື່ອຕົວເລືອກຫຼາຍຕົວກຳນົດເປົ້າໝາຍອົງປະກອບດຽວກັນ. ຮູບແບບໃນແຖວ ມີຄວາມສະເພາະເຈາະຈຶງສູງສຸດ ຖຸມາແມ່ນ ຕົວເລືອກ ໄອດີ(ID), ຕົວເລືອກ ຄລາສ(Class) ແລະ ຕົວເລືອກອົງປະກອບ

ແນະນຳກ່ຽວກັບການພັດທະນາເວັບ ແລະ React.js

JavaScript



JavaScript ເປັນພາສາໂປ່ງແກມທີ່ໃຊ້ສ້າງເອັຟເຟັກແບບໂຕຕອບພາຍໃນເວັບບຣາວເຊີ. ໂຄງສ້າງພື້ນຖານປະກອບດ້ວຍຄໍາສັ່ງ ນິພິດ ແລະ ຕົວດຳເນີນການ.

ແນະນຳກ່ຽວກັບການພັດທະນາເວັບ ແລະ React.js

ຕົວປ່ຽນ, ຕົວດຳເນີນການ ແລະ ນິພິດ:

- ຕົວປ່ຽນ: ປະກາດໂດຍໃຊ້ var, let, or const.
- ຕົວດຳເນີນການ: ເລກຄະນິດ(+, -, *, /), ປຽບທຽບ(==, ===, !=, !==, <, >, <=, >=), ຕັກກະ(&&, ||, !).
- ນິພິດ: ການຮວມກັນຂອງ ຕົວປ່ຽນ ຕົວດຳເນີນການ ແລະ ຄ່າທີ່ສ້າງຜົນຮັບ.



```
1 let x = 5;
2 let y = 6;
3 let z = x + y;
```

ແນະນຳກ່ຽວກັບການພັດທະນາເວັບ ແລະ React.js

ໂຄງສ້າງຂອງການຄວບຄຸມ (ຄໍາສົ່ງ if, loops):

- ຄໍາສົ່ງ if: ໃຊ້ໃນການຮັນໂຄດຕາມເງື່ອນໄຂ. ເຊັ່ນຕົວຢ່າງ: if (condition) { // code }.
- Loops: ໃຊ້ໃນການຮັນໂຄດຊ້າງ. ເຊັ່ນຕົວຢ່າງ: ລວມມີfor, while, and do-while loops.

ແນະນຳກ່ຽວກັບການພັດທະນາເວັບ ແລະ React.js

ໂຄງສ້າງຂອງການຄວບຄຸມ (ຄໍາສັ່ງ if, loops):



```
1  if (hour < 18) {  
2      greeting = "Good day";  
3  }  
4
```

if statements



```
1  for (let i = 0; i < cars.length; i++) {  
2      text += cars[i] + "<br>";  
3  }  
4
```

Loops

ແນະນຳກ່ຽວກັບການພັດທະນາເວັບ ແລະ React.js

Functionsและevents:

- Functions: ບູອກໂຄດ ທີ່ອອກແບບມາເພື່ອດຳເນີນການ ວຽກງານໃດໜີ້ງສະເພາະ ໂດຍ ກຳນົດໃຊ້ຄໍາສໍາຄັນຂອງຝຶກ. ເຊັ່ນຕົວຢ່າງ: ພຶກຂັນmyFunction() { // code }.
- Events: ການດຳເນີນການທີ່ເກີດຂຶ້ນໃນບຮາວເຊີ ເຊັ່ນ: ການຄຣິກ, ການສົ່ງແບບຟອມ ຫຼື ການໂຫຼດໜ້າເພີ. ສາມາດໃຊ້ຕົວຮັບຝຶກເຫດການເພື່ອດຳເນີນການໂຄດເພື່ອຕອບສະໜອງຕໍ່ເຫດການເຫຼົ້ານີ້. ເຊັ່ນຕົວຢ່າງ: element.addEventListener('click', function() { // code });

ແນະນຳກ່ຽວກັບການພັດທະນາເວັບ ແລະ React.js

Functions ແລະ events:

```
1 // Function is called, the return value will end up in x
2 let x = myFunction(4, 3);
3
4 function myFunction(a, b) {
5   // Function returns the product of a and b
6   return a * b;
7 }
8
```

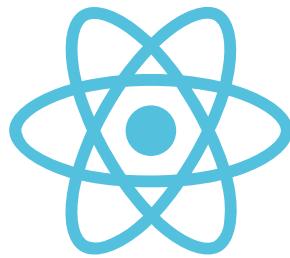
function

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <h1>JavaScript HTML Events</h1>
5 <h2>The onclick Attribute</h2>
6
7 <p>Click the button to display the date.</p>
8 <button onclick="displayDate()">The time is?</button>
9
10 <script>
11   function displayDate() {
12     document.getElementById("demo").innerHTML = Date();
13   }
14 </script>
15
16 <p id="demo"></p>
17
18 </body>
19 </html>
```

events

ແນະນຳກ່ຽວກັບການພັດທະນາເວັບ ແລະ React.js

ແນະນຳກ່ຽວກັບ React



React ເປັນໄລ້ນາລີ JavaScript ສໍາງບັນການສ້າງໜ້າເພີ້ມ ເພື່ອເຊື່ອມຕໍ່ກັບຜູ້ໃຊ້ງານ. ຊ່ວຍໃຫ້ຜູ້ພັດທະນາສາມາດສ້າງ ແອັບພື້ນເຄຊັນເວັບຂະໜາດໃຫຍ່ ທີ່ສາມາດອັບເດດ ແລະ ສະແດງຜົນ ດັ່ງນີ້ມີປະສິດທິພາບເມື່ອມີການປ່ຽນແປງຂໍ້ມູນ.

ແນະນຳກ່ຽວກັບການພັດທະນາເວັບ ແລະ React.js

ການຕັ້ງຄ່າ ໂປ່ງເຈັກ React

- ຕິດຕັ້ງ `create-react-app`: ເປີດ Terminal (macOS) ຫຼື Command Prompt/PowerShell (Windows) ແລ້ວ ຮັນຄໍາສົ່ງດັ່ງດີໄປນີ້:

```
npm install -g create-react-app
```

- ສ້າງໂປ່ງເຈັກໃໝ່ ຂອງ React: ໄປຍັງໄດ້ເຮັກທຳລີ ທີ່ເຮົາຕ້ອງການສ້າງໂປ່ງເຈັກ ແລະ ຮັນ:

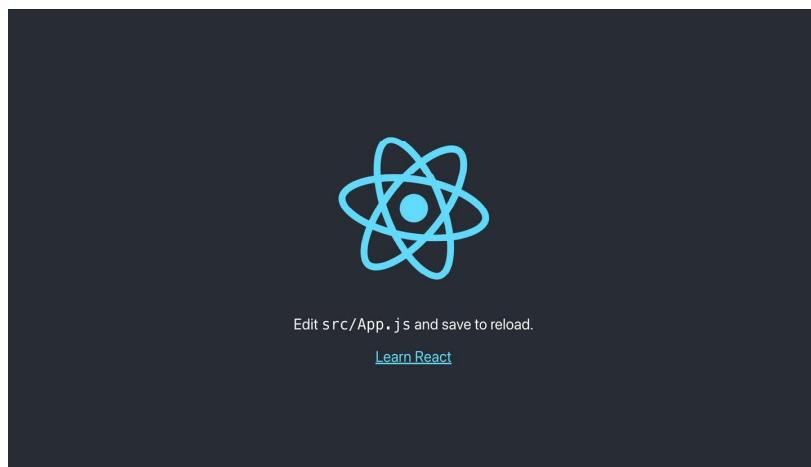
```
npx create-react-app my-app  
cd my-app
```

- ເລີ່ມພັດທະນາໃນເຊີເວີ : ເລີ່ມໃຊ້ຄໍາສົ່ງຕໍ່ໄປນີ້ ເພື່ອເລີ່ມເຂົ້າໄປພັດທະນາໃນເຊີເວີ:

```
npm start
```

Introduction to Web Development and React.js

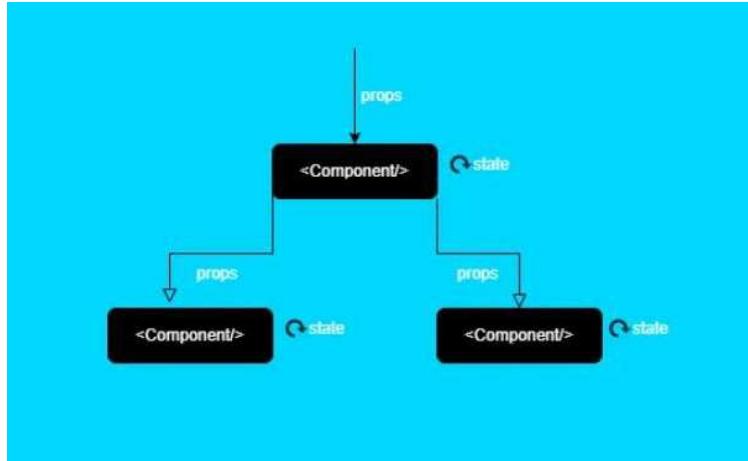
- Open Your React Application: Open a web browser and go to <http://localhost:3000> to see your new React app running.



ສ່ວນປະກອບ ແລະ ຄຸນສົມບັດ

ແນະນຳກ່ຽວກັບສ່ວນປະກອບ ແລະ ຄຸນສົມບັດ

- React components ເປັນສ່ວນສຳຄັນຂອງ ແອັບພື້ເຄຊັນ React.
- Components ສາມາດເຮັດວຽກໄດ້ ຫຼື ຕາມຄລາສ.
- Props ໃຊ້ໃນການສົ່ງຜ່ານຂໍ້ມູນຈາກສ່ວນປະກອບຫຼັກ ໄປຍັງອີງປະກອບຍ່ອຍ.



ສ່ວນປະກອບ ການເຮັດວຽກ

ສ່ວນປະກອບການເຮັດວຽກໃໝ່ JavaScript ທີ່ສົ່ງຄ່າຄົນ JSX.

- ໃນໜາຍ/src/App.js

```
1  function Greeting() {
2      return <h1>Hello, World!</h1>;
3  }
```

1. ສ້າງຟັງຊັ້ນ Greetingຕ້ອຍ <h1>

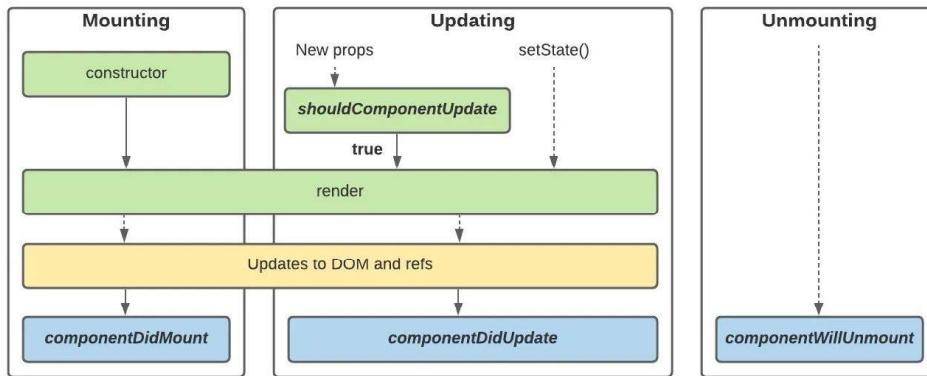
```
1  import "./App.css";
2
3  function Greeting() {
4      return <h1>Hello, World!</h1>;
5  }
6
7  function App() {
8      return (
9          <div className="App">
10              <Greeting></Greeting>
11          </div>
12      );
13  }
14
15 export default App;
```

2. ເອັນໃຊ້ຟັງຊັ້ນ Greetingໃນຟັງຊັ້ນApp

State และ Lifecycle

แนะนำถึง State และ Lifecycle

- State: แนวความคิดหลักใน React สำลับกับคุณสมบัติของชั้นบนแบบเดื่องทางพายในส่วนประกอบ
- Lifecycle: ชุดของวิธีการเขียนว่าในแต่ละขั้นตอนที่แทรกต่างกันของส่วนประกอบ

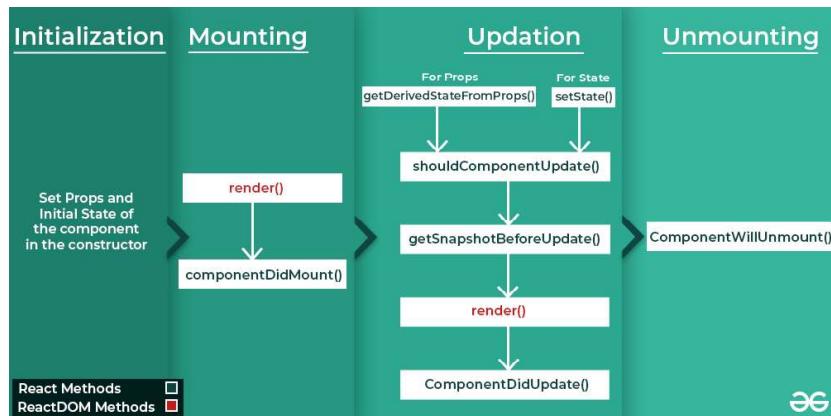


- State เป็นใช้กับการอัปเดตแบบ inline มิภโดยบ่มีการโหลดหน้า ถ้าใช้
- Lifecycle methods ให้กับ component ที่กำหนดตัวเอง

State และ Lifecycle

แนะนำวิธีการ Lifecycle Methods

- Lifecycle methods แม่นวิธีการพิเศษถึงกับส่วนประกอบของ Class ที่เรียกว่า อัปเดตโน้มดโดย React
- อะนุญาตให้เริ่ม เริ่มใช้คำสั่งในเวลาที่กำหนด ของส่วนประกอบ
- Common phases: การติดตั้ง, การอัปเดต และ การทุบของการติดตั้ง



ပါတနောက်

1. HTML သိမ်မာဂာယျ?

- a) High Tech Markup Language
- b) HyperText Markup Language
- c) Home Tool Markup Language
- d) Hyperlink and Text Markup Language

2. HTML tag ၏အားဖြင့်အောင် heading ဘဲဖော်ပဲဆုံး?

- a) <h6>
- b) <h3>
- c) <h1>
- d) <header>

ပါတနောက်

3. CSS သိမ်မာဂာယျ?

- a) Computer Style Sheets
- b) Cascading Style Sheets
- c) Creative Style System
- d) Colorful Style Sheets

4. CSS property ၏အားဖြင့်အောင်စိတ်ကာမ်းစီးပွားရေး?

- a) background-color
- b) font-color
- c) color
- d) text-color

ບົດເຝັກຫັດ

5. React ແມ່ນຫຍັງ?

- a) ຖານຂໍ້ມູນ
- b) JavaScript library ສໍາລັບສ້າງ user interfaces
- c) ເວັບເຊີບເວີ
- d) CSS framework

6. ຄໍາສົ່ງໃດທີ່ໃຊ້ໃນການສ້າງ React application?

- a) npm create react-app myapp
- b) npx create-react-app myapp
- c) npm install react myapp
- d) npx install react-app myapp

ບົດເຝັກຫັດ

7. props ໃນ React ແມ່ນຫຍັງ?

- a) ຄຸນສົມບັດທີ່ຖືກສົ່ງຕໍ່ຈາກ component ແມ່ ໄປຫາ component ລູກ
- b) ຮູບແບບ CSS
- c) ການເຊື່ອມຕໍ່ຖານຂໍ້ມູນ
- d) ຄຸນລັກສະນະ HTML

8. JSX ແມ່ນຫຍັງ?

- a) ພາສາສອບຖານຖານຂໍ້ມູນ (A database query language)
- b) ສ່ອນຂະຫຍາຍໄວຍະກອນສໍາລັບ JavaScript ທີ່ມີລັກສະນະຄ້າຍຄື HTML
- c) CSS preprocessor
- d) ເວັບເຊີບເວີ

ບົດເຟັກຫັດ

9. ວິທີໃດທີ່ໃຊ້ໃນການອັບເດດ state ໃນ functional components?

- a) this.setState()
- b) ພຶ້ງຊັ້ນຕົວຕົ້ງຄ່າສະຖານະ (State setter function) ຈາກ useState hook
- c) updateState()
- d) changeState()

10. ໂດຍຄ່າເລີ່ມຕົ້ນ, ເຊີນເວີພັດທະນາຂອງ React ຈະເຮັດວຽກຢູ່ທີ່ພອດໃດ?

- a) 8080
- b) 3000
- c) 5000
- d) 4000

ມື້ທີ 6: Tailwind CSS ແລະ Component Styling

ການແນະນຳ Utility Classes ຂອງ Tailwind CSS

Tailwind CSS ແມ່ນຫຍັງ?

- Framework CSS ແບບເນື້ນ utility-first
- ແກນທີ່ຈະຊຽນ CSS ທີ່ກຳນົດເອງ, ທ່ານໃຊ້ classes ທີ່ມີຢູ່ແລ້ວ
- ແຕ່ລະ class ເຮັດວຽກສະເພາະຢ່າງດຽວ

Traditional CSS vs Tailwind:

The image shows two side-by-side code snippets in a dark-themed code editor window. Both snippets represent the same button component.

Left (Traditional CSS):

```
/* Traditional CSS */
.button {
  background-color: blue;
  color: white;
  padding: 8px 16px;
  border-radius: 4px;
}
```

Right (Tailwind CSS):

```
/* Tailwind CSS */
<button className="bg-blue-500 text-white px-4 py-2 rounded">
  Click me
</button>
```

ການແນະນຳ Utility Classes ຂອງ Tailwind CSS

ຜົນປະໂຫຍດ:

- ພັດທະນາໄດ້ໄວຂຶ້ນ
- ການອອກແບບທີ່ສອດຄ່ອງກັນ
- ບໍ່ຈໍາເປັນຕິອງຊຽນ CSS ແບບກຳຫນົດເອງ
- ງ່າຍຕໍ່ການອ່ານແລະເຂົ້າໃຈ

ການຕັ້ງຄ່າ Tailwind ໃນໂປຣເຈັກ React ຂອງທ່ານ

1. ຕິດຕັ້ງ Tailwind CSS

```
npm install -D tailwindcss postcss autoprefixer  
npx tailwindcss init -p
```

2. ຕັ້ງຄ່າ tailwind.config.js

```
module.exports = {  
  content: [  
    "./src/**/*.{js,jsx,ts,tsx}",  
  ],  
  theme: {  
    extend: {},  
  },  
  plugins: [],  
}
```

ການຕັ້ງຄ່າ Tailwind ໃນໂປຣເຈັກ React ຂອງທ່ານ

3. ເພີ່ມ Tailwind ເຂົ້າໄປໃນໄຟລ໌ CSS ຂອງທ່ານ (src/index.css)

```
@tailwind base;  
@tailwind components;  
@tailwind utilities;
```

4. ເລີ່ມໃຊ້ Tailwind classes ໃນ components ຂອງທ່ານ!

ການຢືນຢັນ: ເພີ່ມ className="text-blue-500" ໃສ່ອົງປະກອບໃດໜຶ່ງເພື່ອກວດສອບ

ການຈັດຮູບແບບພື້ນຖານ - ສີ, ໄລຍະຫ່າງ, ແລະ Typography

ສີ:

- ສີຕົວອັກສອນ: text-red-500, text-blue-700, text-gray-900
- ສີພື້ນຫຼັງ: bg-red-500, bg-blue-700, bg-gray-100
- ລະດັບສີ: 50 (ອ່ອນທີ່ສຸດ) ຫາ 900 (ເຂັ້ມທີ່ສຸດ)

ໄລຍະຫ່າງ (Padding & Margin):

- Padding: p-4 (ທຸກດ້ານ), px-2 (ແນວອນ), py-3 (ແນວຕັ້ງ)
- Margin: m-4 (ທຸກດ້ານ), mx-auto (ຢູ່ກາງ), mt-6 (ດ້ານເທິງ)
- ຂະໜາດ: 1 = 0.25rem, 2 = 0.5rem, 4 = 1rem, 8 = 2rem

Typography:

- ຂະໜາດຕົວອັກສອນ: text-sm, text-base, text-lg, text-xl, text-2xl
- ນໍ້າໜັກຕົວອັກສອນ: font-light, font-normal, font-bold, font-black
- ການຈັດວາງຕົວອັກສອນ: text-left, text-center, text-right

ການຈັດຮູບແບບພື້ນຖານ - ສີ, ໄລຍະຫ່າງ, ແລະ Typography

ຕົວຢ່າງ

```
<h1 className="text-2xl font-bold text-gray-900 mb-4">
  Welcome to My App
</h1>
<p className="text-base text-gray-600 px-4 py-2">
  This is a paragraph with Tailwind styling.
</p>
```

Layout Classes - Flexbox and

Grid

Flexbox Layout:

- ពួកបានជុំ (Container): flex
- ទិន្នន័យ (Direction): flex-row, flex-col
- ការងារចំណែកថ្មី (Justify content): justify-start, justify-center, justify-between
- ការងារចំណល់រូបរាយការណា (Align items): items-start, items-center, items-end
- ការងារខ្លះ (Flex wrap): flex-wrap, flex nowrap

តើវិញ្ញាបានដឹងទុកដាក់ Flexbox ដើម្បីរាយការណា:

```
    
<div className="flex justify-between items-center">  
  <h1>Title</h1>  
  <button>Action</button>  
</div>
```

Layout Classes - Flexbox and

Grid

Grid Layout:

- ពួកបានជុំ (Container): grid
- ភ័ណ៌ (Columns): grid-cols-1, grid-cols-2, grid-cols-3, grid-cols-4
- ខែងខែង (Gap): gap-2, gap-4, gap-6

តើវិញ្ញាបានដឹងទុកដាក់ Grid:

```
    
<div className="grid grid-cols-3 gap-4">  
  <div>Item 1</div>  
  <div>Item 2</div>  
  <div>Item 3</div>  
</div>
```

ການຈັດຮູບແບບ Components ທີ່ມີຢູ່ແລ້ວຂອງທ່ານ

ກ່ອນໃຊ້ Tailwind (React ທຳມະດາ):

```
function Header() {
  return (
    <div>
      <h1>My Website</h1>
      <nav>
        <a href="/">Home</a>
        <a href="/about">About</a>
      </nav>
    </div>
  );
}
```

ການຈັດຮູບແບບ Components ທີ່ມີຢູ່ແລ້ວຂອງທ່ານ

ຫຼັງຈາກໃຊ້ Tailwind:

```
function Header() {
  return (
    <div className="bg-white shadow-md px-6 py-4">
      <div className="flex justify-between items-center">
        <h1 className="text-xl font-bold text-gray-800">My Website</h1>
        <nav className="flex space-x-4">
          <a href="/" className="text-gray-600 hover:text-blue-500">Home</a>
          <a href="/about" className="text-gray-600 hover:text-blue-500">About</a>
        </nav>
      </div>
    </div>
  );
}
```

ການຈັດຮູບແບບ Component ປຸມ ແລະ Form

Primary Button:



```
<button className="bg-blue-500 hover:bg-blue-600 text-white font-medium py-2 px-4 rounded">  
  Primary Action  
</button>
```

Secondary Button:



```
<button className="bg-gray-200 hover:bg-gray-300 text-gray-800 font-medium py-2 px-4 rounded">  
  Secondary Action  
</button>
```

ການຈັດຮູບແບບ Component ປຸມ ແລະ Form

ການຈັດຮູບແບບ Form Input:



```
<div className="mb-4">  
  <label className="block text-gray-700 text-sm font-medium mb-2">  
    Email Address  
</label>  
  <input  
    type="email"  
    className="w-full px-3 py-2 border border-gray-300 rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500"  
    placeholder="Enter your email"  
  />  
</div>
```

ຕົວຢ່າງຟອມທີ່ສົມບູນ:



```
<form className="bg-white p-6 rounded-lg shadow-md">  
  <h2 className="text-lg font-bold mb-4">Contact Us</h2>  
  {/* Form fields here */}  
</form>
```

Card และ Container Layouts

ส่วนประกอบของ card พื้นฐาน (Basic Card Component):

```
function Card({ title, content }) {
  return (
    <div className="bg-white rounded-lg shadow-md p-6 mb-4">
      <h3 className="text-lg font-semibold mb-2">{title}</h3>
      <p className="text-gray-600">{content}</p>
    </div>
  );
}
```

Card และ Container Layouts

Card และ Image:

```
function ProductCard({ image, name, price }) {
  return (
    <div className="bg-white rounded-lg shadow-md overflow-hidden">
      <img src={image} alt={name} className="w-full h-48 object-cover" />
      <div className="p-4">
        <h3 className="font-semibold text-lg">{name}</h3>
        <p className="text-green-600 font-bold">${price}</p>
      </div>
    </div>
  );
}
```

Card และ Container Layouts

Container Layouts:

```
● ● ●  
// Main container  
<div className="container mx-auto px-4 py-8">  
  {/* Content here */}  
</div>  
  
// Section container  
<section className="bg-gray-50 py-12">  
  <div className="max-w-4xl mx-auto px-6">  
    {/* Section content */}  
  </div>  
</section>
```

Hover และ Interactive States

Hover Effects:

```
● ● ●  
// Button hover  
<button className="bg-blue-500 hover:bg-blue-600 transform hover:scale-105 transition-all">  
  Hover Me  
</button>  
  
// Link hover  
<a href="#" className="text-blue-500 hover:text-blue-700 hover:underline">  
  Hover Link  
</a>  
  
// Card hover  
<div className="bg-white shadow-md hover:shadow-lg transition-shadow cursor-pointer">  
  Hover for shadow effect  
</div>
```

Hover และ Interactive States

ສະຖານະ Focus (ເພື່ອການເຂົ້າເຖິງ):



```
<input className="border border-gray-300 focus:border-blue-500 focus:ring-2 focus:ring-blue-200" />  
<button className="bg-blue-500 focus:outline-none focus:ring-4 focus:ring-blue-300">  
    Accessible Button  
</button>
```

Active States:



```
<button className="bg-blue-500 hover:bg-blue-600 active:bg-blue-700">  
    Click Me  
</button>
```

ສີພື້ນຫຼັງ ແລະ ຂອບ

Transition Classes:

- transition-all: ການປ່ຽນຜ່ານຢ່າງລຽບງ່າຍສໍາລັບທຸກຄຸນສົມບັດ
- transition-colors: ການປ່ຽນຜ່ານສະເພາະສີເທົ່ານັ້ນ
- duration-300: ໄລຍະເວລາການປ່ຽນຜ່ານ
- ease-in-out: ຈັງຫວະການປ່ຽນຜ່ານ

ຕົວຢ່າງການໂຕ້ຕອບທີ່ສົມບູນ:



```
<div className="bg-white p-4 rounded-lg shadow-md hover:shadow-xl transform hover:-translate-y-1 transition-all duration-300 cursor-pointer">  
    Interactive Card  
</div>
```

ສີພື້ນຫຼັງ ແລະ ຂອບ

ວິທີ:



```
// Solid backgrounds
<div className="bg-blue-500">Blue background</div>
<div className="bg-gray-100">Light gray background</div>
<div className="bg-green-50">Very light green</div>

// Gradient backgrounds
<div className="bg-gradient-to-r from-blue-500 to-purple-600">
  Gradient background
</div>
```

ສີພື້ນຫຼັງ ແລະ ຂອບ

Border Styling:



```
// Basic borders
<div className="border">Default border</div>
<div className="border-2 border-blue-500">Colored thick border</div>
<div className="border-t-4 border-red-500">Top border only</div>

// Border radius
<div className="rounded">Small rounded corners</div>
<div className="rounded-lg">Large rounded corners</div>
<div className="rounded-full">Fully rounded (circle/pill)</div>

// Combining borders and backgrounds
<div className="bg-white border-2 border-gray-200 rounded-lg p-4">
  Card with border and background
</div>
```

ສີພື້ນຫຼັງ ແລະ ຂອບ

Shadow Effects:



```
<div className="shadow-sm">Small shadow</div>
<div className="shadow-md">Medium shadow</div>
<div className="shadow-lg">Large shadow</div>
<div className="shadow-xl">Extra large shadow</div>
```

ສີພື້ນຫຼັງ ແລະ ຂອບ

ຕົວຢ່າງພາກປະຕິບັດ:



```
function Alert({ type, message }) {
  const styles = {
    success: "bg-green-50 border-green-200 text-green-800",
    error: "bg-red-50 border-red-200 text-red-800",
    warning: "bg-yellow-50 border-yellow-200 text-yellow-800"
  };

  return (
    <div className={`p-4 border rounded-lg ${styles[type]}`}>
      {message}
    </div>
  );
}
```

ການຈັດຮູບແບບຂໍ້ຄວາມ ແລະ ນາ້ມໍ່ໜັກຕົວອັກສອນ

ຂະໜາດຝ່ອນ:



```
<p className="text-xs">Extra small text</p>          /* 12px */  
<p className="text-sm">Small text</p>                /* 14px */  
<p className="text-base">Base text</p>              /* 16px */  
<p className="text-lg">Large text</p>                /* 18px */  
<p className="text-xl">Extra large text</p>           /* 20px */  
<p className="text-2xl">2X large text</p>            /* 24px */  
<p className="text-3xl">3X large text</p>           /* 30px */
```

ການຈັດຮູບແບບຂໍ້ຄວາມ ແລະ ນາ້ມໍ່ໜັກຕົວອັກສອນ

ສີຕົວອັກສອນ ແລະ ການຈັດວາງ



```
<p className="text-gray-500">Gray text</p>  
<p className="text-blue-600">Blue text</p>  
<p className="text-red-500">Red text</p>  
  
<p className="text-left">Left aligned</p>  
<p className="text-center">Center aligned</p>  
<p className="text-right">Right aligned</p>
```

ການຈັດຮູບແບບຂໍ້ຄວາມ ແລະ ນາ້ມໍ່າກຕົວອັກສອນ

Text Decoration:



```
<p className="underline">Underlined text</p>
<p className="line-through">Strikethrough text</p>
<p className="uppercase">UPPERCASE TEXT</p>
<p className="lowercase">lowercase text</p>
<p className="capitalize">Capitalized Text</p>
```

ການຈັດຮູບແບບຂໍ້ຄວາມ ແລະ ນາ້ມໍ່າກຕົວອັກສອນ

ຕົວຢ່າງລາດັບຊັ້ນຂອງ Typography



```
<article className="max-w-2xl">
  <h1 className="text-3xl font-bold text-gray-900 mb-4">Article Title</h1>
  <h2 className="text-xl font-semibold text-gray-800 mb-3">Section Heading</h2>
  <p className="text-base text-gray-700 leading-relaxed mb-4">
    This is the main paragraph text with good readability.
  </p>
  <p className="text-sm text-gray-500">
    This is smaller supporting text or metadata.
  </p>
</article>
```

ການໃຊ້ Spacing ແລະ Padding

ການເຂົ້າໃຈຂະໜາດ Spacing ຂອງ Tailwind:

- 1 = 0.25rem (4px)
- 2 = 0.5rem (8px)
- 3 = 0.75rem (12px)
- 4 = 1rem (16px)
- 6 = 1.5rem (24px)
- 8 = 2rem (32px)
- 12 = 3rem (48px)

ການໃຊ້ Spacing ແລະ Padding

Padding Classes:



```
// All sides
<div className="p-4">Padding on all sides</div>

// Horizontal and vertical
<div className="px-4 py-2">Different horizontal and vertical</div>

// Individual sides
<div className="pt-4 pr-3 pb-2 pl-1">Individual side padding</div>
```

ການໃຊ້ Spacing ແລະ Padding

Margin Classes:



```
// All sides margin  
<div className="m-4">Margin on all sides</div>  
  
// Auto centering  
<div className="mx-auto">Centered horizontally</div>  
  
// Negative margins (for overlaps)  
<div className="-mt-4">Negative top margin</div>
```

ບົດເຝັກຫັດ

1. Tailwind CSS ແມ່ນ CSS framework ປະເພດໃດ?

- a) ແບບອີງໃສ່ອົງປະກອບ (Component-based)
- b) ແບບເນັ້ນການໃຊ້ງານ (Utility-first)
- c) ແບບອີງໃສ່ແມ່ນແບບ (Template-based)
- d) ແບບອີງໃສ່ຕາໜ່າງ (Grid-based)

2. Tailwind ໃຊ້ຄໍາສັ່ງໄຕໃດໃນການກຳນົດຕາໜັງສີເປັນສີໜ້າ?

- a) blue-text
- b) text-blue-500
- c) color-blue
- d) font-blue

ບົດເຝັກຫັດ

3. class p-4 ເຮັດໜ້າທີ່ຫຍຸງໃນ Tailwind?

- a) ກຳນົດ padding ໃຫ້ເປັນ 4px
- b) ກຳນົດ padding ໃຫ້ເປັນ 1rem (16px)
- c) ກຳນົດຕຳແໜ່ງໃຫ້ເປັນ 4
- d) ກຳນົດໜ້າໃຫ້ເປັນ 4

4. class Tailwind ໂດຍໃຫ້ກຳນົດໃຫ້ເນື້ອຫາຢູ່ທາງກາງຕາມແວວນອນ?

- a) center
- b) mx-auto
- c) horizontal-center
- d) text-center

ບົດເຝັກຫັດ

5. ການສ້າງ flex container ໃນ Tailwind ຕ້ອງໃຊ້ຄຳສັ່ງໄດ້?

- a) display-flex
- b) flex
- c) flexbox
- d) d-flex

6. ໃນ Tailwind, class ທີ່ໃຊ້ສ້າງ grid ທີ່ມີ 3 ຖັນ ແມ່ນໂຕໄດ້?

- a) grid-3
- b) columns-3
- c) grid-cols-3
- d) col-3

ບົດເຝັກຫັດ

7. hover: prefix ใน Tailwind ແມ່ນເຮັດໜຳທີ່ຫຍັງ?

- a) ນຳໃຊ້ຮູບແບບເນື້ອວາງເນື້າໄສ
- b) ສ້າງອະນິມເຊັນແບບ hover
- c) ເຮັດໃຫ້ອົງປະກອບສາມາດ hover ໄດ້
- d) ລົບລ້າງຜົນກະທົບຂອງ hover

8. ຄໍາສັ່ງທີ່ໃຊ້ເພີ່ມເງິນໃຫ້ກັບອົງປະກອບແມ່ນໂຕໃດ?

- a) shadow
- b) box-shadow
- c) drop-shadow
- d) shadow-effect

ບົດເຝັກຫັດ

9. ໃນ Tailwind, ທ່ານສາມາດເຮັດໃຫ້ຂໍ້ຄວາມເປັນ bold ໂດ້ໂດຍໃຊ້ class ໃດ?

- a) text-bold
- b) font-bold
- c) bold
- d) weight-bold

10. ຄໍາສັ່ງທີ່ໃຊ້ສ້າງຂອບມືນ (rounded corners) ແມ່ນຂໍ້ໃດ?

- a) round
- b) border-round
- c) rounded
- d) corner-round

ມີຫີ 7: ການເຊື່ອມຕໍ່ React ກັບ Backend API ຂອງ ທ່ານ

ການຮຽກໃຊ້ API ດ້ວຍ Fetch ໃນ React

Fetch API ແມ່ນຫຍັງ?

- ພົງຊັນ JavaScript ທີ່ມີຢູ່ໃນຕົວສໍາລັບການຮ້ອງຂໍ HTTP
- ເປັນຕົວແທນທີ່ທັນສະໄໝຂອງ XMLHttpRequest
- ສ່ົງຄົນ Promises (ໃຊ້ໄດ້ດີກັບ async/await)
- ອອງຮັບໃນທຸກໆ browser ທີ່ທັນສະໄໝ



```
fetch(url, options)
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

ການຮຽກໃຊ້ API ດ້ວຍ Fetch ໃນ React

ການໃຊ້ Async/Await (ວິທີທີ່ແນະນຳ):

```
async function fetchData() {
  try {
    const response = await fetch('http://localhost:5000/api/posts');
    const data = await response.json();
    console.log(data);
  } catch (error) {
    console.error('Error:', error);
  }
}
```

ການຮ້ອງຂໍແບບ GET (Fetch Data):

```
const response = await fetch('http://localhost:5000/api/posts');
```

ການຮຽກໃຊ້ API ດ້ວຍ Fetch ໃນ React

ການຮ້ອງຂໍແບບ POST (ສົ່ງຂໍ້ມູນ):

```
const response = await fetch('http://localhost:5000/api/posts', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({
    title: 'New Post',
    content: 'Post content'
  })
});
```

ການຮ້ອງຂໍແບບ POST (ສົ່ງຂໍ້ມູນ):

- Content-Type: application/json - ສໍາລັບການສົ່ງຂໍ້ມູນແບບ JSON
- Authorization: Bearer \${token} - ສໍາລັບການຮ້ອງຂໍທີ່ຕ້ອງການການຍືນຍັນຕົວຕົນ (authenticated requests)

useEffect Hook សំលែកការណ៍ឱ្យខ្លួន

useEffect មែនម្បោរ?

- បែង React Hook សំលែកការបង្កើត side effects
- នៅពេលក្នុងការ component ពិនិត្យ render ឡើង
- ឈ្មោះសំលែកការរូបភាព API, subscriptions, timers
- មានហេតុផ្ទុកជាធម្ម័យ componentDidMount, componentDidUpdate, และវីនិច្ឆ័យ.

Basic useEffect Syntax:

```
import { useEffect, useState } from 'react';

function MyComponent() {
  useEffect(() => {
    // Side effect code here
    console.log('Component mounted or updated');
  });

  return <div>My Component</div>;
}
```

useEffect Hook សំលែកការណ៍ឱ្យខ្លួន

useEffect ដែលត្រូវការពិនិត្យ Dependency Array:

```
function PostsList() {
  const [posts, setPosts] = useState([]);

  // Run only once when component mounts
  useEffect(() => {
    fetchPosts();
  }, []); // Empty dependency array

  async function fetchPosts() {
    try {
      const response = await fetch('http://localhost:5000/api/posts');
      const data = await response.json();
      setPosts(data.posts);
    } catch (error) {
      console.error('Error fetching posts:', error);
    }
  }

  return (
    <div>
      {posts.map(post => (
        <div key={post._id}>{post.title}</div>
      )));
    </div>
  );
}
```

useEffect Hook សំលែកវាងការបង្កើត

បំពី Dependencies (ឡើងរបស់ខ្លួនពេលទូទាត់ render):

```
useEffect(() => {
  // Runs after every render
});
```

ឬវាត្រូវបានដោឡូចនៅពេលដែឡូច (ឡើងរបស់ខ្លួនពេលដែឡូច mount):

```
useEffect(() => {
  // Runs once when component mounts
}, []);
```

ឡើងរបស់ខ្លួនពេលដែឡូចនៅពេលដែឡូចនៅពេលដែឡូចនៅពេលដែឡូច

```
useEffect(() => {
  // Runs when userId changes
  fetchUserData(userId);
}, [userId]);
```

useEffect Hook សំលែកវាងការបង្កើត

Cleanup Function:

```
useEffect(() => {
  const timer = setInterval(() => {
    console.log('Timer tick');
  }, 1000);

  // Cleanup function
  return () => {
    clearInterval(timer);
  };
}, []);
```

ການສະແດງຂໍ້ມູນ Backend ໃນ Components

ຕົວຢ່າງທີ່ສົມບູນ - ລາຍຊື້ໄພສ:

```
const [posts, setPosts] = useState([]);

useEffect(() => {
  fetchPosts();
}, []);

async function fetchPosts() {
  try {
    const response = await fetch('http://localhost:5000/api/posts');
    const data = await response.json();
    setPosts(data.posts || []);
  } catch (error) {
    console.error('Error:', error);
  }
}
```

ການສະແດງຂໍ້ມູນ Backend ໃນ Components

```
return (
  <div className="max-w-4xl mx-auto p-6">
    <h1 className="text-2xl font-bold mb-6">All Posts</h1>
    <div className="space-y-4">
      {posts.map(post => (
        <div key={post._id} className="bg-white p-6 rounded-lg shadow-md">
          <h2 className="text-xl font-semibold mb-2">{post.title}</h2>
          <p className="text-gray-600 mb-4">{post.content}</p>
          <div className="text-sm text-gray-500">
            By: {post.author?.name || 'Unknown'}
          </div>
        </div>
      )));
    </div>
  );
}
```

ສະຖານະການກຳລັງໂຫຼດ (Loading States) ແລະ Feedback ຈາກຜູ້ໃຊ້ (User Feedback)

ເປັນຫຍັງ Loading States ຈຶ່ງສໍາຄັນ:

- ຜູ້ໃຊ້ຈະເປັນຕ້ອງຮູ້ວ່າກຳລັງມີຫຍັງເກີດຂຶ້ນ
- ປ້ອງກັນຄວາມສັບສົນໃນລະຫວ່າງການຮ້ອງຂໍ API
- ປັບປຸງປະສິດທິພາບທີ່ຮັບຮູ້ໄດ້
- ປະສົບການຜູ້ໃຊ້ທີ່ດີຂຶ້ນ

ສະຖານະການກຳລັງໂຫຼດ (Loading States) ແລະ Feedback ຈາກຜູ້ໃຊ້ (User Feedback)

Basic Loading State:

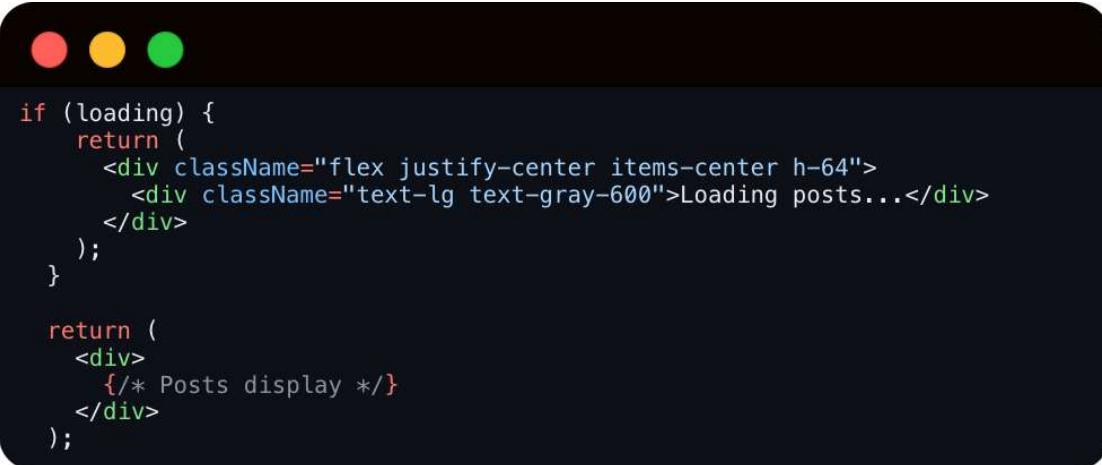


```
const [posts, setPosts] = useState([]);
const [loading, setLoading] = useState(true);

useEffect(() => {
  fetchPosts();
}, []);

async function fetchPosts() {
  try {
    setLoading(true);
    const response = await fetch('http://localhost:5000/api/posts');
    const data = await response.json();
    setPosts(data.posts || []);
  } catch (error) {
    console.error('Error:', error);
  } finally {
    setLoading(false);
  }
}
```

ສະຖານະການກຳລັງໂຫຼດ (Loading States) ແລະ Feedback ຈາກຜູ້ໃຊ້ (User Feedback)



```
if (loading) {
  return (
    <div className="flex justify-center items-center h-64">
      <div className="text-lg text-gray-600">Loading posts...</div>
    </div>
  );
}

return (
  <div>
    {/* Posts display */}
  </div>
);
```

ການຈັດການຂໍຜິດພາດໃນການຮ້ອງຂໍ API

ປະເພດຂອງຂໍຜິດພາດ:

- ຂໍຜິດພາດຂອງເຄືອຂ່າຍ - ເຊີບເວີລື່ມ, ບໍ່ມີອິນເຕີເນັດ
- ຂໍຜິດພາດ HTTP - 404, 500, 401, ແລະ ອື່ນໆ
- ຂໍຜິດພາດຂອງຂໍ້ມູນ - ຮູບແບບການຕອບກັບບໍ່ຖືກຕ້ອງ
- ຂໍຜິດພາດຂອງການຢືນຢັນຕົວຕົນ - Token ໝົດອາຍຸ, ບໍ່ໄດ້ຮັບອະນຸຍາດ

ການຈັດການຂໍຜິດພາດໃນການຮ້ອງຂໍ API

ຮູບແບບການຈັດການຂໍຜິດພາດແບບສົມບູນ:

```
const [posts, setPosts] = useState([]);
const [loading, setLoading] = useState(true);
const [error, setError] = useState(null);

useEffect(() => {
  fetchPosts();
}, []);
```

ການຈັດການຂໍຜິດພາດໃນການຮ້ອງຂໍ API

Complete Error Handling Pattern:

```
async function fetchPosts() {
  try {
    setLoading(true);
    setError(null); // Clear previous errors

    const response = await fetch('http://localhost:5000/api/posts');

    // Check if response is ok
    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`);
    }

    const data = await response.json();

    // Validate data structure
    if (!data.posts) {
      throw new Error('Invalid response format');
    }

    setPosts(data.posts);
  } catch (error) {
    console.error('Error fetching posts:', error);
    setError(error.message);
  } finally {
    setLoading(false);
  }
}
```

ການຈັດການຂໍຜິດພາດໃນການຮ້ອງຂໍ API



```
if (error) {
  return (
    <div className="bg-red-50 border border-red-200 rounded-lg p-4">
      <h3 className="text-red-800 font-medium">Error Loading Posts</h3>
      <p className="text-red-600 mt-1">{error}</p>
      <button
        onClick={fetchPosts}
        className="mt-3 bg-red-500 text-white px-4 py-2 rounded hover:bg-red-600"
      >
        Try Again
      </button>
    </div>
  );
}
```

ການສ່ົງຂໍ້ມູນໄປທາ Backend (POST Requests)

ການສ່ົງຂໍ້ມູນໄປທາ Backend (POST Requests)



```
async function createPost(postData) {
  try {
    const response = await fetch('http://localhost:5000/api/posts', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(postData)
    });

    if (!response.ok) {
      throw new Error('Failed to create post');
    }

    const data = await response.json();
    return data;
  } catch (error) {
    console.error('Error creating post:', error);
    throw error;
  }
}
```

ການສ່ົງຂໍ້ມູນໄປທາ Backend (POST Requests)

POST ດ້ວຍ Authentication:

```
async function createPost(postData) {
  try {
    const token = localStorage.getItem('authToken');

    const response = await fetch('http://localhost:5000/api/posts', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${token}`
      },
      body: JSON.stringify(postData)
    });

    if (!response.ok) {
      const errorData = await response.json();
      throw new Error(errorData.message || 'Failed to create post');
    }

    return await response.json();
  } catch (error) {
    throw error;
  }
}
```

ສ່ົງຂໍ້ມູນໄປ backend (PUT Requests)

PUT Request (ການອັບເດດ):

```
async function updatePost(postId, updateData) {
  try {
    const token = localStorage.getItem('authToken');

    const response = await fetch(`http://localhost:5000/api/posts/${postId}`, {
      method: 'PUT',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${token}`
      },
      body: JSON.stringify(updateData)
    });

    if (!response.ok) {
      throw new Error('Failed to update post');
    }

    return await response.json();
  } catch (error) {
    throw error;
  }
}
```

ການສ່ົງຂໍ້ມູນໄປທາ Backend (DELETE Requests)

DELETE Request: ຄໍາຮ້ອງຂໍເພື່ອລຶບຂໍ້ມູນ



```
async function deletePost(postId) {
  try {
    const token = localStorage.getItem('authToken');

    const response = await fetch(`http://localhost:5000/api/posts/${postId}`, {
      method: 'DELETE',
      headers: {
        'Authorization': `Bearer ${token}`
      }
    });

    if (!response.ok) {
      throw new Error('Failed to delete post');
    }

    return await response.json();
  } catch (error) {
    throw error;
  }
}
```

ການສ່ົງຂໍ້ມູນປະເພດຕ່າງໆ

ຂໍ້ມູນແບບຟອມ (ສໍາລັບການອັບໂຫຼດໄຟລ്)



```
async function uploadFile(file) {
  const formData = new FormData();
  formData.append('file', file);
  formData.append('description', 'My file');

  const response = await fetch('http://localhost:5000/api/upload', {
    method: 'POST',
    body: formData // Don't set Content-Type header for FormData
  });
}
```

ບົດເຝັກຫັດ

1. Fetch API ແມ່ນຫຍັງ?

- a) ການເລື່ອນໄຟລ໌ຈາກຄອມບົວເຕີ
- b) ການສ້າງຄໍາວ່ອງຂໍ HTTP ໄປທາເຊີເວີ
- c) ການດຶງຂໍ້ມູນຈາກ localStorage
- d) ການດາວໂຫຼດຮູບພາບ

2. React hook ໂດຍໃດທີ່ມັກໃຊ້ໃນການເຮັ້ນ API?

- a) useState
- b) useEffect
- c) useContext
- d) useReducer

ບົດເຝັກຫັດ

3. [] ໃນ useEffect ໝາຍຄວາມວ່າແນວໃດ?

- a) ຜົນກະທົບເຮັດວຽກທຸກຄັ້ງທີ່ render
- b) ຜົນກະທົບເຮັດວຽກພຽງຄັ້ງດຽວເມື່ອ component ຕິກສ້າງຂຶ້ນ
- c) ຜົນກະທົບບໍ່ມີວັນເຮັດວຽກ
- d) ຜົນກະທົບເຮັດວຽກເມື່ອ component ຕິກເອົາອອກ

4. HTTP method ໄດ້ທີ່ໃຊ້ສໍາລັບສ້າງຂໍ້ມູນໃໝ່?

- a) GET
- b) POST
- c) PUT
- d) DELETE

ບົດເຝັກຫັດ

5. header ໄດທີ່ຈະເປັນຕ້ອງສິ່ງໄປເມື່ອຕ້ອງສິ່ງຂໍ້ມູນ JSON data?

- a) Accept: application/json
- b) Content-Type: application/json
- c) Authorization: Bearer token
- d) User-Agent: React

6. ວິທີໃດທີ່ໃຊ້ຈັດການ loading state ໃນ React?

- a) ບໍລິສິນໃຈ
- b) ໃຊ້ useState ເພື່ອຕິດຕາມສະຖານະການໂຫຼດ
- c) ສະແດງສະເພາະຂໍ້ຜິດພາດ
- d) ໃຊ້ console.log

ບົດເຝັກຫັດ

7. ວິທີໃດຖືກຕ້ອງທີ່ສຸດໃນການຈັດການຂໍ້ຜິດພາດເມື່ອເຮົາ fetch requests?

- a) ບໍລິສິນໃຈຂໍ້ຜິດພາດ
- b) ໃຊ້ try-catch blocks ຮ່ວມກັບ async/await
- c) ພຽງແຕ່ສະແດງຂໍ້ຜິດພາດໃນ console
- d) ໂຫຼດໜ້າຄືນໃໝ່

8. HTTP method ໂຕໃດທີ່ໃຊ້ໃນການອັບເດດຂໍ້ມູນທີ່ມີຢູ່?

- a) POST
- b) GET
- c) PUT
- d) DELETE

ບົດເຟັກຫັດ

9. ສິ່ງທີ່ຄວນຮັດໃນຂະນະທີ່ການຮ້ອງຂໍ API ກໍາລັງດຳເນີນການປູ່?

- a) ບໍ່ສະແດງຫຍຸງເລີຍ
- b) ສະແດງສົນຍາລັກບອກການໂຫຼດ
- c) ປົດການໃຊ້ງານແອບທັງໝົດ
- d) ປຸຽນເສັ້ນທາງໄປໜ້າອື່ນ

10. ວິທີໃດທີ່ລວມ authentication tokens ເຂົ້າໃນ API requests?

- a) ໃນ URL
- b) ໃນ Authorization header
- c) ໃນ request body
- d) ໃນ localStorage ເພື່ອນັ້ນ

ມື້ທີ 8: ການໂຕ້ຕອບກັບຜູ້ໃຊ້ & dynamic features

ການຈັດການຟອມໃນ React

ເຫດຜົນທີ່ການຈັດການຟອມມີຄວາມສໍາຄັນ

- ຈໍາເປັນຕົວເກັບກຳ, ຢືນຢັນຄວາມຖືກຕ້ອງ, ແລະປະມວນຜົນຂໍ້ມູນທີ່ຜູ້ໃຊ້ປ້ອນເຂົ້າມາ.
- React ມີຮູບແບບສະເພາະສໍາລັບການຈັດການຟອມຢ່າງມີປະສິດທິພາບ.

ການຈັດການຟອມໃນ React

Controlled Components (ແນະນຳ):

- ຄອບຄຸມຄ່າຂອງ Input ດ້ວຍ state
- ແຫ່ງຂໍ້ມູນດຽວ
- ງ່າຍຕໍ່ການກວດສອບ ແລະ ອີເຊັດ

```
const [name, setName] = useState('');  
  
<input  
    value={name}  
    onChange={(e) => setName(e.target.value)}  
/>
```

Controlled Components และ ການຈັດການ Input

Controlled Components ແມ່ນຫຍັງ?

- ອີງປະກອບຟອມທີ່ຄ່າຂອງມັນຖືກຄວບຄຸມໂດຍ React state
- React ກາຍເປັນ "ແຫຼ້ງຂໍ້ມູນຫຼັກ"
- ການປ່ຽນແປງທັງໝົດຕ້ອງຜ່ານຕົວຈັດການເຫດການ (event handlers) ຂອງ React

ການຈັດການ Input ປະເພດຕ່າງໆ

Text Inputs:

- <input type="text">: ຂໍ້ຄວາມທົ່ວໄປ
- <input type="email">: ອີເມວ (ກວດສອບຮູບແບບອີເມວ)
- <input type="password">: ລະຫັດຜ່ານ (ຂໍ້ຄວາມທີ່ຖືກຊ່ອນ)
- <textarea>: ຂໍ້ຄວາມຫຼາຍບັນຫັດ

Controlled Components และ ການຈັດການ Input

Selection Inputs:

- <select>: ແນູ້ເລືອກ (Dropdown menus)
- <input type="radio">: ທາງເລືອກດຽວ (Single choice)
- <input type="checkbox">: ຫຼາຍທາງເລືອກ (Multiple choices)

```
// For checkbox, use 'checked' instead of 'value'  
<input  
  type="checkbox"  
  checked={formData.agree}  
  onChange={(e) => setFormData(prev => ({  
    ...prev,  
    agree: e.target.checked  
}))}  
/>
```

ການເພີ່ມຝັງຊື້ນຄົ້ນຫາ

ເປັນຫຍັງຝັງຊື້ນຄົ້ນຫາຈຶ່ງສໍາຄັນ:

- ຜູ້ໃຊ້ຕ້ອງການຊອກຫາເນື້ອໃນທີ່ສະເພາະຢ່າງວ່ອໄວ
- ປັບປຸງປະສົບການຂອງຜູ້ໃຊ້ຢ່າງຫຼວງຫຼາຍ
- ຈໍາເປັນສໍາລັບແອັບທີ່ມີຂໍ້ມູນຈຳນວນຫຼາຍ

ການຕິດຕັ້ງຝັງຊື້ນຄົ້ນຫາແບບພື້ນຖານ

- ເພີ່ມຊ່ອງຂໍ້ມູນ (input field) ສໍາລັບຄົ້ນຫາ
- ເກັບຄຳຄົ້ນຫາໄວ້ໃນ state
- ກອງຂໍ້ມູນໂດຍອີງໄສ່ຄຳຄົ້ນຫາ
- ສະແດງຜົນການກອງ

ການກອງ ແລະ ການຈັດຮຽງ

ການກອງ ແລະ ການຄົ້ນຫາ

- Search (ການຄົ້ນຫາ): ຊອກຫາສິນຄ້າທີ່ມີຂໍ້ຄວາມສະເພາະ
- Filter (ການກອງ): ສະແດງ/ເຊື່ອງສິນຄ້າໂດຍອີງໄສ່ປະເພດ ຫຼື ຄຸນສົມບັດ
- Sort (ການຈັດຮຽງ): ປ່ຽນລຳດັບການສະແດງຜົນຂອງສິນຄ້າ

ປະເພດການກອງທົ່ວໄປ:

- Category filters: ຕົວກອງຕາມໝວດໝູ່ (ແບບເລືອກລົງ ຫຼື ປຸ່ມກົດ)
- Date range filters: ຕົວກອງຕາມຊ່ວງວັນທີ
- Status filters: ຕົວກອງຕາມສະຖານະ (ເຊັ່ນ: ເຜີຍແຜ່ແລ້ວ, ແບບຮ່າງ, ແລະອື່ນໆ)
- User/author filters: ຕົວກອງຕາມຜູ້ໃຊ້/ຜູ້ຂຽນ

ການກອງ ແລະ ການຈັດຮຽງ

Pattern ໃນການນຳໃຊ້ Filter:

- 1.Create filter state: ສ້າງ state ສໍາລັບເກີບຄ່າຂອງຕົວກອງ
- 2.Apply filters to data: ນຳໃຊ້ຕົວກອງກັບຂໍ້ມູນ
- 3.Update UI based on filtered results: ອັບເດດ UI ໂດຍອີງໃສ່ຜົນການກອງ
- 4.Provide "clear filters" option: ຈັດຫາທາງເລືອກ "ລຶບຕົວກອງ" (ເພື່ອລຶບລ້າງຕົວກອງທີ່ເລືອກໄວ້)

ການກອງ ແລະ ການຈັດຮຽງ

Sorting Implementation:

```
const [sortConfig, setSortConfig] = useState({  
    field: 'createdAt',  
    direction: 'desc'  
});  
  
const sortedPosts = useMemo(() => {  
    return [...posts].sort((a, b) => {  
        if (a[sortConfig.field] < b[sortConfig.field]) {  
            return sortConfig.direction === 'asc' ? -1 : 1;  
        }  
        if (a[sortConfig.field] > b[sortConfig.field]) {  
            return sortConfig.direction === 'asc' ? 1 : -1;  
        }  
        return 0;  
    });  
, [posts, sortConfig]);
```

ການອັບໂຫຼດຮູບພາບພ້ອມກັບການສະແດງຕົວຢ່າງ

ເປັນຫຍັງການອັບໂຫຼດຮູບພາບຈຶ່ງເປັນສິ່ງສໍາຄັນ

- Visual content engages users: ເນື້ອໃນແບບເບິ່ງເຫັນເຮັດໃຫ້ຜູ້ໃຊ້ສົນໃຈ
- Essential for many types of apps: ຈະເປັນສໍາລັບແຮ້ບພລິເຄຊັ້ນຫຼາຍປະເພດ
- Improves content quality: ປັບປຸງຄຸນນະພາບຂອງເນື້ອໃນ

ຂັ້ນຕອນພື້ນຖານໃນການອັບໂຫຼດ:

1. User selects image file: ຜູ້ໃຊ້ເລືອກໄຟລ໌ຮູບພາບ
2. Show preview immediately: ສະແດງຕົວຢ່າງທັນທີ
3. Upload to server: ອັບໂຫຼດຂຶ້ນເຊີບເວີ
4. Display success/error feedback: ສະແດງຄໍາຕອບເມື່ອສໍາເລັດ ຫຼື ແກັດຂໍ້ຜິດພາດ

ການອັບໂຫຼດຮູບພາບພ້ອມກັບການສະແດງຕົວຢ່າງ

File Input Handling:

```
function handleImageSelect(e) {
  const file = e.target.files[0];
  if (file) {
    setSelectedImage(file);
    // Create preview URL
    const url = URL.createObjectURL(file);
    setPreviewUrl(url);
  }
}
```

ການລຶບ ແລະ ການແກ້ໄຂ

CRUD Operations ໃນ React:

- Create (ສ້າງ): ເພີ່ມລາຍການໃໝ່
- Read (ອ່ານ): ສະແດງລາຍການ
- Update (ອັບເດດ): ແກ້ໄຂລາຍການທີ່ມີຢູ່ແລ້ວ
- Delete (ລຶບ): ລຶບລາຍການ

ຂັ້ນຕອນການລຶບ (Delete Operation Flow):

1. User clicks delete button: ຜູ້ໃຊ້ກິດປຸ່ມລຶບ
2. Show confirmation dialog: ສະແດງກ່ອງຂໍ້ຄວາມເພື່ອຢືນຢັນ
3. If confirmed, call delete API: ຖ້າຢືນຢັນ, ໃຫ້ເອີ້ນໃຊ້ delete API
4. Update UI immediately: ອັບເດດ UI ທັນທີ
5. Show success/error message: ສະແດງຂໍ້ຄວາມສໍາເລັດ ຫຼື ຂໍຜິດພາດ

ການລຶບ ແລະ ການແກ້ໄຂ

Confirmation Patterns:

- Simple confirm dialog: ກ່ອງຂໍ້ຄວາມຢືນຢັນແບບງ່າຍ
- Modal with details: ປ້ອບອັບ (modal) ພ້ອມລາຍລະອຽດ
- Inline confirmation: ການຢືນຢັນແບບຝ່າຍ (inline)
- Undo functionality: ພັງຊັ້ນຍົກເລີກການກະທຳ (undo)

ຂັ້ນຕອນການແກ້ໄຂ (Edit Operation Flow):

1. User clicks edit button: ຜູ້ໃຊ້ກິດປຸ່ມແກ້ໄຂ
2. Show edit form: ສະແດງຟອມແກ້ໄຂ (ແບບປ້ອບອັບ ຫຼື ແບບຝ່າຍ)
3. Pre-populate with current data: ດຶງຂໍ້ມູນປັດຈຸບັນມາໃສ່ຟອມກ່ອນ
4. Handle form submission: ຈັດການການສົ່ງຟອມ
5. Update UI with new data: ອັບເດດ UI ດ້ວຍຂໍ້ມູນໃໝ່

UI Update Strategies:

ການອັບເດດແບບທັນທີ (Optimistic Updates)

- Update UI immediately: ອັບເດດ UI ທັນທີ
- Rollback if API call fails: ຖອຍຫຼັງຖ້າການເລື່ອນໃຊ້ API ຜິດພາດ
- Better user experience: ປະສົບການຜູ້ໃຊ້ດີຂຶ້ນ

ການອັບເດດແບບລໍຖ້າ (Pessimistic Updates)

- Wait for API response: ລໍຖ້າການຕອບກັບຈາກ API
- Then update UI: ແລ້ວຈຶ່ງອັບເດດ UI
- More reliable but slower feeling: ເຊື້ອຖືໄດ້ຫຼາຍກວ່າ ແຕ່ຮູ້ສຶກຊ້າກວ່າ

ການຈັດການກັບຂໍຜິດພາດ (Error Handling)

- Clear error messages: ຂໍຄວາມແຈ້ງເຕືອນຂໍຜິດພາດທີ່ຊັດເຈນ
- Retry options: ຫາງເລືອກໃນການລອງໃໝ່
- Rollback failed operations: ຖອຍຫຼັງການດຳເນີນງານທີ່ຜິດພາດ
- User-friendly language: ພາສາທີ່ເຂົ້າໃຈງ່າຍສໍາລັບຜູ້ໃຊ້

UI Update Strategies:

State Management:

```
function handleDelete(id) {
    // Remove from UI immediately
    setPosts(prev => prev.filter(post => post.id !== id));

    // Call API
    deletePost(id).catch(error => {
        // Restore on error
        setPosts(originalPosts);
        showError('Delete failed');
    });
}
```

ການເຂົ້າສູ່ລະບົບ/ອອກຈາກລະບົບ

ອົງປະກອບ UI ຂອງການຢືນຢັນຕົວຕົນ

- Login form: ພຼມເຂົ້າສູ່ລະບົບ
- Registration form: ພຼມລົງທະບຽນ
- Password reset: ຕັ້ງລະຫັດຜ່ານໃໝ່
- User profile menu: ແນວໂປຣໄຟລີ່ຜູ້ໃຊ້
- Protected routes: ເສັ້ນທາງທີ່ຖືກປົກປ້ອງ

ສິ່ງສໍາຄັນຂອງພຼມເຂົ້າສູ່ລະບົບ (Login Form Essentials)

- Email/username field: ຊ່ອງໃສ່ອີເມວ/ຊື່ຜູ້ໃຊ້
- Password field: ຊ່ອງໃສ່ລະຫັດຜ່ານ
- Remember me option: ທາງເລືອກ "ຈຶ່ງຂ້ອຍ"
- Forgot password link: ລົງ "ລືມລະຫັດຜ່ານ"
- Clear error messages: ຂໍຄວາມແຈ້ງເຕືອນຂໍຜິດພາດທີ່ຊັດເຈນ

ການເຂົ້າສູ່ລະບົບ/ອອກຈາກລະບົບ

ການກວດສອບຄວາມຖືກຕ້ອງຂອງພຼມ (Form Validation)

- Email format validation: ການກວດສອບຮູບແບບອີເມວ
- Password requirements: ຂໍກຳນົດຂອງລະຫັດຜ່ານ
- Real-time feedback: ການແຈ້ງເຕືອນແບບທັນທີ
- Server-side error handling: ການຈັດການຂໍຜິດພາດຢູ່ຝັ້ງເຊີນເວີ

ການເຂົ້າສູ່ລະບົບ/ອອກຈາກລະບົບ

ການຈັດການສະຖານະການຢືນຢັນຕົວຕົນ

```
const [user, setUser] = useState(null);
const [loading, setLoading] = useState(true);

function login(email, password) {
    // Call login API
    // Store token and user data
    // Redirect to dashboard
}

function logout() {
    // Clear stored data
    // Redirect to login
    setUser(null);
}
```

ການເຂົ້າສູ່ລະບົບ/ອອກຈາກລະບົບ

ການຈັດການໂທເຄັນ

- Store in localStorage/sessionStorage: ເກັບໄວ້ໃນ localStorage ຫຼື sessionStorage
- Include in API requests: ແມບໃສ່ໃນການຮ້ອງຂໍ API
- Handle token expiration: ຈັດການເມື່ອໂທເຄັນໝົດອາຍຸ
- Refresh tokens automatically: ສ້າງໂທເຄັນໃໝ່ໂດຍອັດຕະໂນມັດ

Protected Routes:

- Check authentication before rendering: ກວດສອບການຢືນຢັນຕົວຕົນກ່ອນທີ່ຈະສະແດງຜົນ
- Redirect to login if not authenticated: ປ່ຽນເສັ້ນທາງໄປຫາໜ້າເຂົ້າສູ່ລະບົບຖ້າບໍ່ໄດ້ຮັບການຢືນຢັນຕົວຕົນ
- Show loading while checking auth status: ສະແດງສະຖານະກຳລັງໂຫຼດໃນຂະນະທີ່ກຳລັງກວດສອບສະຖານະການຢືນຢັນຕົວຕົນ

ການເຂົ້າສູ່ລະບົບ/ອອກຈາກລະບົບ

ອົງປະກອບເມນູຜູ້ໃຊ້

- User avatar/initials: ຮູບໂປຣໄຟລ് ຫຼືຕົວອັກສອນຍໍ້ຂອງຜູ້ໃຊ້
- Dropdown with options: ແນູນແບບເລື່ອນລົງພ້ອມທາງເລືອກຕ່າງໆ
- Profile link: ລົງໄປຫາໂປຣໄຟລ്
- Settings link: ລົງໄປຫາການຕັ້ງຄ່າ
- Logout button: ປຸ່ມອອກຈາກລະບົບ

Navigation ລະຫວ່າງ Views ທີ່ແຕກຕ່າງກັນ

ປະເພດຂອງ Navigation:

- Tab navigation: ການນຳທາງແບບແຖນ (ໃຊ້ແຖນເພື່ອສະຫຼັບລະຫວ່າງໜ້າຕ່າງໆ)
- Side navigation: ການນຳທາງດ້ານຂ້າງ (ແນູນຢູ່ດ້ານຂ້າງຂອງໜ້າຈຳ)
- Breadcrumb navigation: ການນຳທາງແບບ Breadcrumb (ສະແດງເສັ້ນທາງທີ່ຜູ້ໃຊ້ໄດ້ຜ່ານມາ)
- Multi-step forms: ພອມຫຼາຍຂັ້ນຕອນ (ຟອມທີ່ແບ່ງອອກເປັນຂັ້ນຕອນ)
- Modal navigation: ການນຳທາງແບບ Modal (ໃຊ້ປ້ອບອັບເພື່ອສະແດງຂໍ້ມູນ)

Tab Navigation:

- Clear visual indicators: ຕົວຊີ້ບອກທີ່ເບິ່ງເຫັນໄດ້ຢ່າງຊັດເຈນ
- Active state styling: ການຈັດຮູບແບບໃຫ້ສະແດງສະຖານະທີ່ກຳລັງໃຊ້ງານຢູ່
- Keyboard navigation support: ອອງຮັບການນຳທາງດ້ວຍແປ້ນພິມ
- Mobile-friendly design: ການອອກແບບທີ່ໃຊ້ງານຈ່າຍໃນມືຖື

Navigation ละหัว Views ที่แตกต่างกัน

State-Based Navigation:



```
const [activeTab, setActiveTab] = useState('posts');

// Render different components based on active tab
{activeTab === 'posts' && <PostsList />}
{activeTab === 'create' && <CreatePost />}
{activeTab === 'profile' && <UserProfile />}
```

ปิดฝึกหัด

1. controlled components ແມ່ນຫຍັງໃນ React?

- a) Components ທີ່ຄວບຄຸມອົງປະກອບອື່ນງ.
- b) Components ຂອງພອມທີ່ມີຄ່າຕ່າງໆທີ່ກຳຄວບຄຸມໂດຍ React state.
- c) Components ທີ່ບໍ່ສາມາດປ່ຽນແປງຄ່າໄດ້
- d) Components ທີ່ບໍ່ມີ state

2. Event handler ທີ່ໃຊ້ສໍາລັບການສົ່ງພອມແມ່ນໄດ້ຂຶ້ນ?

- a) onClick
- b) onChange
- c) onSubmit
- d) onInput

ບົດເຝັກຫັດ

3. CRUD ຫຍ້ມາຈາກຫຍັງ?

- a) Create, Read, Update, Delete
- b) Copy, Remove, Undo, Delete
- c) Create, Remove, Upload, Download
- d) Control, Read, Update, Display

4. ຈະອັບໂທລດຟາຍໃນ React ຈຶ່ງໃດ?

- a) ໃຊ້ `<input type="file">`
- b) ໃຊ້ພຽງແຕ່ການລາກແລະວາງ (drag and drop) ເທົ່ານັ້ນ
- c) ໃຊ້ component ໄຟລີ້ເສດ
- d) ບໍ່ສາມາດອັບໂຫຼດໄຟລີໃນ React ຕັ້ງ

ບົດເຝັກຫັດ

5. ການອັບເດດແບບທັນທີ (Optimistic Updates) ແມ່ນຫຍັງ?

- a) ດີ positiveຢູ່ສະເໜີ
- b) ອັບເດດ UI ທັນທີກ່ອນທີ່ການເອັ້ນໃຊ້ API ຈະສໍາເລັດ.
- c) ອັບເດດຂຶ້ນທີ່ສໍາເລັດ
- d) ບໍ່ອັບເດດ UI

6. ເພື່ອຍືນຢັນການລຶບຂໍ້ມູນ, ມານຄວນໃຊ້ວິທີການໃດ?

- a) ບໍ່ຍືນຢັນຫຍັງ, ລຶບໂລດ
- b) ໂຊ dialog ເພື່ອຍືນຢັນ
- c) ຕ້ອງການການປ້ອນລະຫັດອີກຄັ້ງ
- d) ສິບອັດຕະໂນມັດຫລັງຈາກ 5 ວິ

ບົດເຝັກຫັດ

7. ຈຸດປະສົງຂອງການກວດສອບຄວາມຖືກຕ້ອງຂອງຟອມ (Form Validation) ແມ່ນຫຍັງ?

- a) ເພື່ອເຮັດໃຫ້ຟອມຍາວຂຶ້ນ
- b) ເພື່ອຮັບປະກັນຄຸນນະພາບຂອງຂໍ້ມູນ ແລະ ປະສົບການຂອງຜູ້ໃຊ້
- c) ເພື່ອເຮັດໃຫ້ຜູ້ໃຊ້ຊ້າລົງ
- d) ເພື່ອເຮັດໃຫ້ການພັດທະນາຍາກຂຶ້ນ

8. ການກວດສອບຄວາມຖືກຕ້ອງຢູ່ຝັ້ງ client (Client-side validation) ຄວນເຮັດຕອນໃດ?

- a) ບໍ່ຕ້ອງເຮັດ
- b) ຕ້ອງ submit form ເທົ່ານັ້ນ
- c) ຕອນຜູ້ໃຊ້ພິມໄລດ
- d) ສະເພາະຕອນທີ່ server ກວດໃຫ້

ບົດເຝັກຫັດ

9. ການນຳໃຊ້ການຊອກຫາ (Search Functionality)

- a) ໃຊ້ແຕ່ການຄົ້ນຫາຢູ່ຝັ້ງ server-side ຕະຫຼອດ
- b) ອີງໃສ່ Local data ໂດຍອີງຕາມຂໍ້ມູນທີ່ຕ້ອງການຄົ້ນຫາ
- c) ໃຊ້ Google search
- d) ພຶ້ງຊັ້ນ Search ເຮັດບໍ່ໄດ້ໃນ React

10. ເພື່ອຈັດການສະຖານະການບືນຍືນເວົ້າຕົນ (Authentication State), ອີທີ່ທີ່ແນະນຳແມ່ນວິທີໃດ?

- a) ໃຊ້ localStorage ເທົ່ານັ້ນ
- b) ໃຊ້ React state ແລະ localStorage/sessionStorage
- c) ໃຊ້ cookies ເທົ່ານັ້ນ
- d) ຢ່າເວັບສະຖານະການບືນຍືນເວົ້າຕົນ

មិះទី 9: ប័ណ្ណុន និង ភាសការង្វឹងខ្លួន (Polish & User Experience)

ការអនុវត្ត Loading States របស់អ្នកប្រើប្រាស់

គឺជាប្រព័ន្ធឌីជីថលដែលអ្នកប្រើប្រាស់

- Users need to know something is happening: ដូចជាអ្នកការង្វឹងការងារនៅក្នុងការងារ។
- Prevents confusion and frustration: ប៉ុន្មានការងារនៅក្នុងការងារនៅក្នុងការងារ។
- Improves perceived performance: ប៉ុន្មានការងារនៅក្នុងការងារនៅក្នុងការងារ។
- Shows the app is working, not broken: សម្រាប់អ្នកប្រើប្រាស់ដូចជាអ្នកការង្វឹងការងារនៅក្នុងការងារនៅក្នុងការងារ។

ການເພີ່ມ Loading States ແບບມືອາຊີບ

Basic Loading Indicator:



```
{loading ? <div>Loading...</div> : <PostsList posts={posts} />}
```

Basic Loading Indicator:



```
function LoadingSpinner() {
  return (
    <div className="flex justify-center items-center">
      <div className="animate-spin rounded-full h-8 w-8 border-b-2 border-blue-500"></div>
    </div>
  );
}
```

ການເພີ່ມ Loading States ແບບມືອາຊີບ

Basic Loading Indicator:

- Shows placeholder content structure: ສະແດງໂຄງສ້າງເນື້ອໃນທີ່ເປັນ Placeholder.
- Matches actual content layout: ກົງກັບໂຄງຮ່າງເນື້ອຫາຕົວຈິງ.
- Better than blank screens: ດີກວ່າໜ້າຈໍ່ຫວ່າງເປົ່າ.
- Users understand what's coming: ຜູ້ໃຊ້ເຂົ້າໃຈວ່າກໍາລັງຈະມີຫຍັງມາ.



```
<button disabled={loading} className={loading ? 'opacity-50' : ''}>
  {loading ? 'Saving...' : 'Save Post'}
</button>
```

ການກວດສອບຄວາມຖືກຕ້ອງຂອງຟອມ ແລະ ການຕອບຮັບກັບຜູ້ໃຊ້

ການກວດສອບຄວາມຖືກຕ້ອງຢູ່ຝຶ່ງ client

- Immediate feedback: ການຕອບຮັບແບບທັນທີ
- No server round-trip needed: ບໍ່ຈໍາເປັນຕ້ອງສ່ວນຂໍ້ມູນໄປ-ກັບເຊີບເວີ
- Better user experience: ປະສົບການຜູ້ໃຊ້ທີ່ດີກວ່າ
- Basic format checking: ການກວດສອບຮູບແບບພື້ນຖານ

ການກວດສອບຄວາມຖືກຕ້ອງຢູ່ຝຶ່ງເຊີບເວີ

- Authoritative validation: ການກວດສອບທີ່ໜ້າເຊື້ອຖືໄດ້
- Security considerations: ການພິຈາລະນາດ້ານຄວາມປອດໄພ
- Business rule enforcement: ການບັງຄັບໃຊ້ກົດລະບຽບທາງທຸລະກົດ
- Database constraint checking: ການກວດສອບຂໍ້ຈຳກັດຂອງຖານຂໍ້ມູນ

ໄລຍະເວລາການກວດສອບຄວາມຖືກຕ້ອງ

ເມື່ອສ່ົງຟອມ (ແບບພື້ນຖານ)

```
function validateForm(data) {  
  const errors = {};  
  if (!data.email) errors.email = 'Email is required';  
  if (!data.password) errors.password = 'Password is required';  
  return errors;  
}
```

ການກວດສອບຄວາມຖືກຕ້ອງແບບທັນທີ (ຂັ້ນສູງ)

- ກວດສອບເມື່ອຜູ້ໃຊ້ພິມ.
- ລົບຂໍ້ຄວາມຜິດພາດອອກທັນທີເມື່ອມີການແກ້ໄຂ.
- ຢ່າສະແດງຂໍ້ຄວາມຜິດພາດແບບຮຸນແຮງເກີນໄປ.
- ສ້າງຄວາມສົມດູນລະຫວ່າງການຊ່ວຍເຫຼືອແລະ ການສ້າງຄວາມລຳຄານ.

ການສ້າງ Animation ທີ່ລຽບງ່າຍດ້ວຍ Tailwind

ຄວາມສໍາຄັນຂອງ Animation

- Provide visual feedback: ໃຫ້ຄໍາຕອບແບບເບິ່ງເຫັນໄດ້.
- Guide user attention: ນຳພາຄວາມສົນໃຈຂອງຜູ້ໃຊ້.
- Make interactions feel responsive: ເຮັດໃຫ້ການໃຊ້ງານຮູ້ສຶກວ່ອງໄວ.
- Add personality to your app: ເພີ່ມບຸກຄະລິກໃຫ້ກັບແຮ້ບຂອງທ່ານ.

Animations ທີ່ມີຢູ່ໃນຕົວ (ຂອງ Tailwind)

- animate-spin: ການໝູນວຽນແບບບໍ່ຢຸດຢັ້ງ.
- animate-ping: ແມ່ດຸດທີ່ມີການເຕັ້ນເປັນຈັງຫວະ.
- animate-pulse: ການປ່ຽນແປງຄວາມໂປ່ງໃສ (opacity) ແບບສິ້ນສະເທືອນ.
- animate-bounce: ການເຄື່ອນໄຫວແບບກະໂດດ.

ການສ້າງ Animation ທີ່ລຽບງ່າຍດ້ວຍ Tailwind

Transition Classes:

- transition-all: ເຮັດໃຫ້ທຸກຄຸນສົມບັດມີການເຄື່ອນໄຫວທີ່ລຽບງ່າຍ.
- transition-colors: ເຮັດໃຫ້ການປ່ຽນແປງສີເທົ່ານັ້ນມີການເຄື່ອນໄຫວ.
- transition-transform: ເຮັດໃຫ້ການປ່ຽນແປງຮູບຮ່າງ (transform) ເທົ່ານັ້ນມີການເຄື່ອນໄຫວ.
- duration-300: ກໍານົດໄລຍະເວລາ 300 ມີລືວິນາທີ.
- ease-in-out: ເຮັດໃຫ້ຈັງຫວະການເຄື່ອນໄຫວລຽບງ່າຍ, ເລີ່ມຕິ້ນຊ້າແລະຈົບລົງຊ້າ.

Hover Animations:



```
<button className="transform hover:scale-105 transition-transform duration-200">  
  Hover Me  
</button>
```

ສະຖານະຫວ່າງເປົ້າ ແລະ ເນື້ອໃນສໍາຮອງ (Placeholder Content)

Empty States (ສະຖານະຫວ່າງເປົ້າ)

- Screens shown when there's no data: ໜ້າຈຳທີ່ສະແດງອອກມາເນື້ອບໍ່ມີຂໍ້ມູນ.
- First-time user experience: ປະສົບການຂອງຜູ້ໃຊ້ໃນຄັ້ງທຳອິດ.
- Search with no results: ການຄົ້ນຫາທີ່ບໍ່ມີຜົນລັບ.
- Deleted or removed content: ເນື້ອໃນທີ່ຖືກລຶບ ຫຼືເອົາອອກໄປ.

ຄວາມສໍາຄັນຂອງ Empty States

- Prevent confusion: ປ້ອງກັນຄວາມສັບສົນ.
- Guide user actions: ນຳພາການກະທຳຂອງຜູ້ໃຊ້.
- Onboard new users: ຊ່ວຍເຫຼືອຜູ້ໃຊ້ໃໝ່ໃຫ້ເລີ່ມຕົ້ນໃຊ້ງານ.
- Maintain engagement: ຮັກສາຄວາມສົນໃຈ.

ປະເພດຂອງສະຖານະຫວ່າງເປົ້າ

"No Data Yet" (ຍັງບໍ່ມີຂໍ້ມູນເທົ່ອ)

- First time using the app: ເປັນການໃຊ້ແອັບຄັ້ງທຳອິດ.
- Encouraging message: ຂໍ້ຄວາມທີ່ໃຫ້ກໍາລັງໃຈ.
- Clear call to action: ການກະທຳທີ່ຊັດເຈນ.
- Show what's possible: ສະແດງໃຫ້ເຫັນສິ່ງທີ່ສາມາດເຮັດໄດ້.

"No Search Results" (ບໍ່ພົບຜົນການຄົ້ນຫາ)

- Acknowledge the search: ຮັບຮູ້ວ່າໄດ້ມີການຄົ້ນຫາ.
- Suggest alternatives: ແນະນຳຫາງເລືອກອື່ນ.
- Offer to broaden search: ສະເໜີໃຫ້ຂະຫຍາຍຂອບເຂດການຄົ້ນຫາ.
- Check spelling suggestions: ແນະນຳໃຫ້ກວດສອບການສະກິດຄໍາ.

ປະເພດຂອງສະຖານະຫວ່າງເປົ້າ

Error State (ສະຖານະຂໍຜິດພາດ)

- Something went wrong: ເກີດຂໍຜິດພາດບາງຢ່າງ
- Clear explanation: ຄໍາອະທິບາຍທີ່ຊັດເຈນ
- Recovery actions: ການປະຕິບັດເພື່ອແກ້ໄຂ
- Try again option: ຫາງເລືອກໃນການລອງອີກຄັ້ງ

Empty State Examples:

```
function EmptyPosts() {  
  return (  
    <div className="text-center py-12">  
      <div className="text-6xl mb-4">No posts yet</div>  
      <h3 className="text-lg font-medium text-gray-900 mb-2">  
        No posts yet  
      </h3>  
      <p className="text-gray-600 mb-6">  
        Get started by creating your first post  
      </p>  
      <button className="bg-blue-500 text-white px-6 py-2 rounded">  
        Create First Post  
      </button>  
    </div>  
  );  
}
```

ການໃຊ້ສີ ແລະ ຄວາມສອດຄ່ອງຫວ່າງດ້ານສາຍຕາ

ຄວາມສໍາຄັນຂອງການໃຊ້ສີ

- Creates emotional response: ສ້າງການຕອບສະໜອງຫວ່າງດ້ານອາລີມ
- Establishes brand identity: ສ້າງເອກະລັກຂອງ brand
- Guides user attention: ນຳພາຄວາມສົນໃຈຂອງຜູ້ໃຊ້
- Improves usability: ປັບປຸງຄວາມສາມາດໃນການໃຊ້ງານ

ຈົດຕະວິທະຍາຂອງສີ

- Blue: ຄວາມໄວ້ວາງໃຈ, ເປັນມືອາຊີບ, ສະຫງົບ.
- Green: ຄວາມສໍາເລັດ, ທຳມະຊາດ, ການເຕີບໂຕ.
- Red: ອັນຕະລາຍ, ຄວາມຮົບດ່ວນ, ຄວາມຮ້ອນແຮງ.
- Yellow: ຄໍາເຕືອນ, ພະລັງງານ, ແວ່ດີ.
- Gray: ເປັນກາງ, ສະຫງ່າ, ໝັ້ນຄົງ.

ລະບົບສີຂອງ Tailwind

ລະບົບສີເທິາ (ພື້ນຖານ)

- gray-50 to gray-900: ຈາກສີເທິາອ່ອນສຸດເຖິງສີເທິາເຂັ້ມສູດ.
- Use for text, borders, backgrounds: ໃຊ້ສຳລັບຂໍ້ຄວາມ, ເສັ້ນຂອບ, ພື້ນຫຼັງ.
- Creates hierarchy without color: ສ້າງລຳດັບຊັ້ນໂດຍບໍ່ຕ້ອງໃຊ້ສີອື່ນ.

ສີຂອງຍື້ຫໍ້

- blue-500: ການກະທຳຫຼັກ (primary actions).
- green-500: ສະຖານະສຳເລັດ (success states).
- red-500: ຂໍຜິດພາດ, ການກະທຳທີ່ສ້າງຄວາມເສຍຫາຍ (errors, destructive actions).
- yellow-500: ຄໍາເຕືອນ (warnings).

ຮບແບບການໃຊ້ສີ

ສີໂຕໝັງສີ

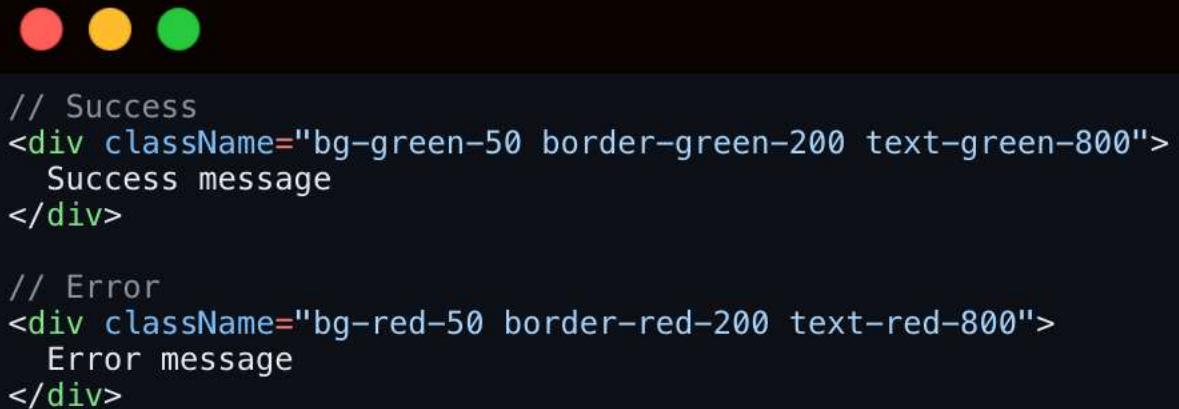
- Primary text: text-gray-900: ຂໍຄວາມຫຼັກ (ສີເທິາເຂັ້ມສູດ).
- Secondary text: text-gray-600: ຂໍຄວາມຮອງ (ສີເທິາກາງ).
- Muted text: text-gray-500: ຂໍຄວາມຈາງ (ສີເທິາອ່ອນກວ່າ).
- Links: text-blue-600: ການເຊື້ອມຕໍ່ (ສີຟ້າເຂັ້ມ).

ສີພື້ນຫຼັງ

- Page background: bg-gray-50: ພື້ນຫຼັງຂອງໜ້າ (ສີເທິາອ່ອນ).
- Card background: bg-white: ພື້ນຫຼັງຂອງ card (ສີຂາວ).
- Active background: bg-blue-50: ພື້ນຫຼັງທີ່ກຳລັງເຮັດວຽກ (ສີຟ້າອ່ອນ).
- Hover background: bg-gray-100: ພື້ນຫຼັງເມື່ອເອົາເມົ້າໄປຊື້ (ສີເທິາອ່ອນກວ່າ).

ຮບແບບການໃຊ້ສີ

ສີທີ່ມີຄວາມໝາຍ



The screenshot shows a mobile application interface with three colored dots at the top (red, yellow, green). Below them, there are two `<div>` elements with specific CSS classes:

```
// Success
<div className="bg-green-50 border-green-200 text-green-800">
  Success message
</div>

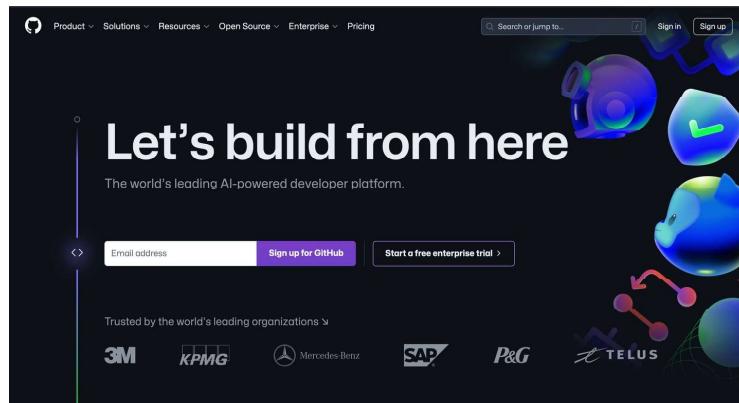
// Error
<div className="bg-red-50 border-red-200 text-red-800">
  Error message
</div>
```

ນີ້ທີ 10: ໂຄງການສຸດທ້າຍ ແລະ ການຮຽນຮູ້ໃນ
ອະນາຄົດ

ການນຳໃຊ້ໂຄງການເຂົ້າສູ່ລະບົບຈິງ ແລະ ໂຄງການສຸດທ້າຍ

ແນະນຳ GitHub

GitHub ແມ່ນແພັນັກຝອມເວັບສໍາລັບການຄວບຄຸມເວີຊັ້ນ ແລະ ການຮັດວຽກຮ່ວມກັນກັບນັກພັດທະນາຄົມອື່ນ ຊ່ວຍໃຫ້ນັກພັດທະນາສາມາດຈັດເກັບ, ຈັດການ ແລະ ຕິດຕາມການປຽນແປງໂຄດຂອງຕົນເອງໄດ້.

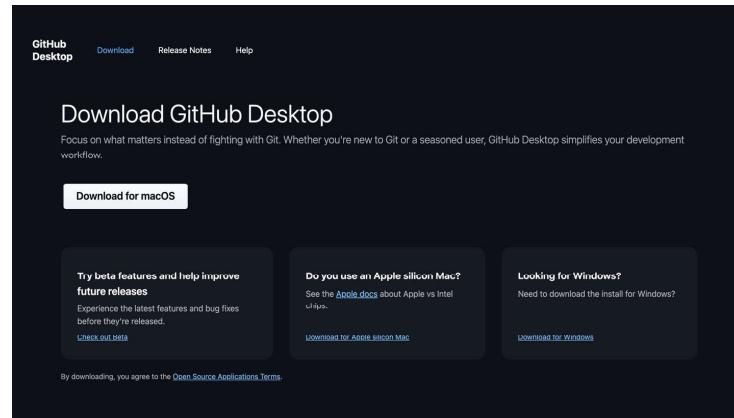


<https://github.com/>

ການນຳໃຊ້ໂຄງການເຂົ້າສູ່ລະບົບຈິງ ແລະ ໂຄງການສຸດທ້າຍ

GitHub Desktop

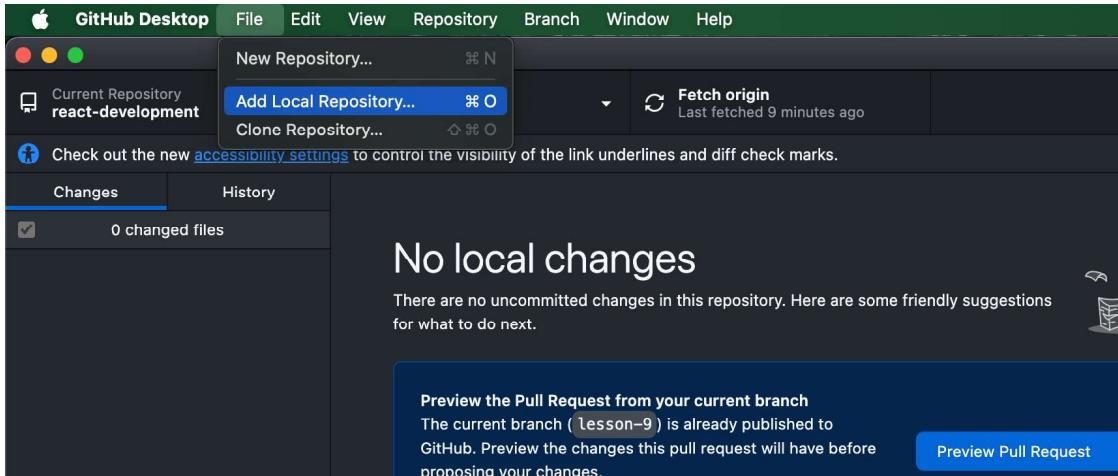
GitHub Desktop ແມ່ນເອັບພື້ນເຄີຊັ້ນທີ່ໃຊ້ງານງ່າຍ ຊ່ວຍໃຫ້ນັກພັດທະນາສາມາດຕອບໄຕກັບ GitHub ໂດຍໃຊ້ໜ້າຕ່າງແບບກອນາຟົກແທນການຂຽນແບບ Command line.



<https://desktop.github.com/download/>

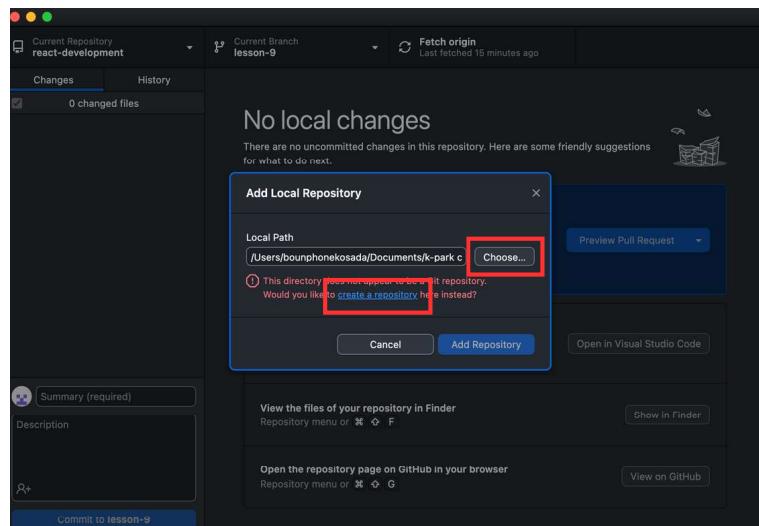
ການນຳໃຊ້ໂຄງການເຂົ້າສູ່ລະບົບຈິງ ແລະ ໂຄງການສຸດທ້າຍ

ກົດ “Add Local Repository...”.



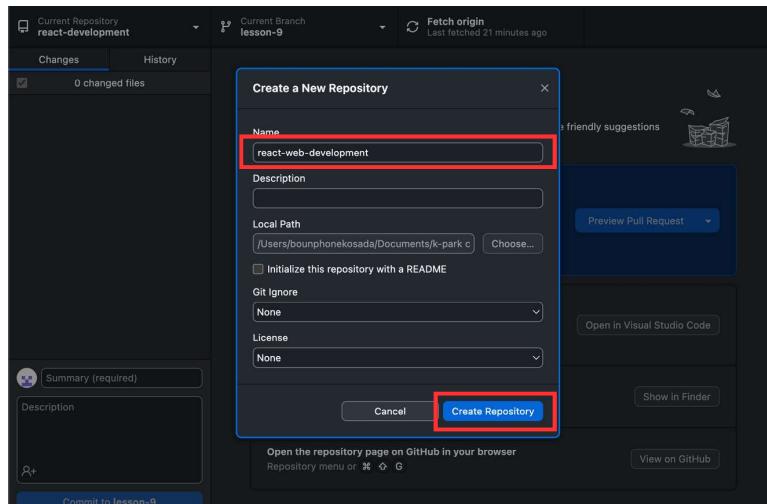
ການນຳໃຊ້ໂຄງການເຂົ້າສູ່ລະບົບຈິງ ແລະ ໂຄງການສຸດທ້າຍ

ກົດ “Choose...” ເພື່ອເລືອກຕຳແໜ່ງທີ່ຈະຜິດຕັ້ງ project ລົງໃນເຄື່ອງ ແລະ ກົດ “create a repository”.



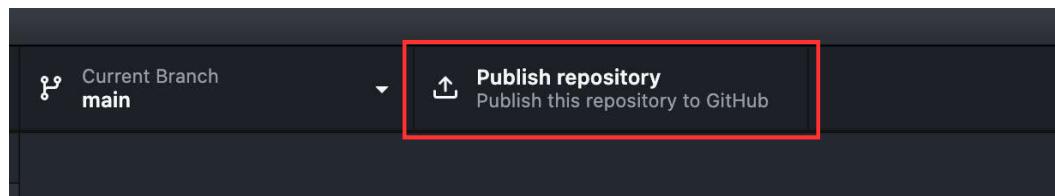
ການນຳໃຊ້ໂຄງການເຂົ້າສູ່ລະບົບຈິງ ແລະ ໂຄງການສຸດທ້າຍ

ພິມຊື່ຂອງ project ໃນຊ່ອງ “Name” ແລະ ກົດ “Create Repository”.



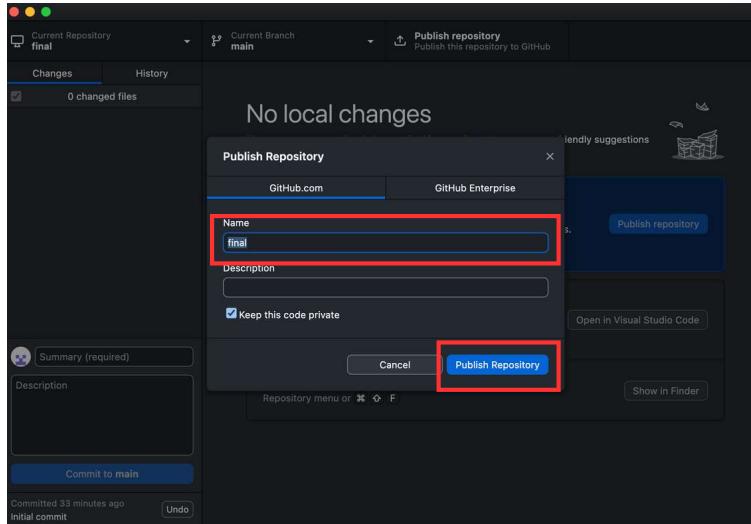
ການນຳໃຊ້ໂຄງການເຂົ້າສູ່ລະບົບຈິງ ແລະ ໂຄງການສຸດທ້າຍ

ກົດ “Publish repository” ເພື່ອສ້າງ project repository ໄປເກັບໄວ້ໃນ GitHub.



ການນຳໃຊ້ໂຄງການເຂົ້າສູ່ລະບົບຈິງ ແລະ ໂຄງການສຸດທ້າຍ

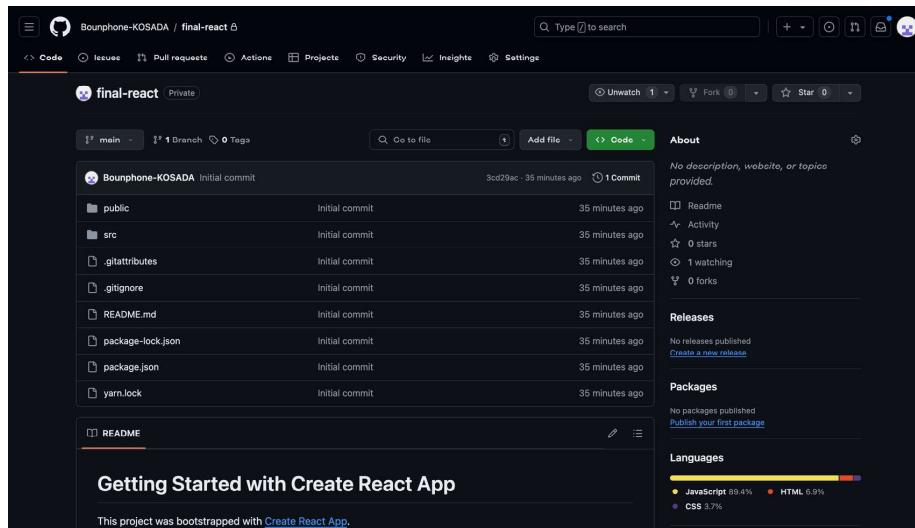
ພິມຊື່ຂອງ repository ແລະ ກົດປຸ່ມ “Publish Repository”.



ການນຳໃຊ້ໂຄງການເຂົ້າສູ່ລະບົບຈິງ ແລະ ໂຄງການສຸດທ້າຍ

GitHub

ຫຼັງຈາກນີ້ ຈະເຫັນ repository ໃນບັນຊີ GitHub.



ການນຳໃຊ້ໂຄງການເຂົ້າສູ່ລະບົບຈິງ ແລະ ໂຄງການສຸດທ້າຍ

ແນະນຳການ Deployment ກັບ Vercel

Vercel ແມ່ນແພວດົບອມຄຣາວສໍາລັບການ deployment ແບບ static ແລະ serverless. ເປັນທີ່ນີ້ຍືມໂດຍສະເພາະສໍາລັບ front-end ແລະ ການສ້າງ static site.

ຄຸນສົມບັດທີ່ສໍາຄັນ:

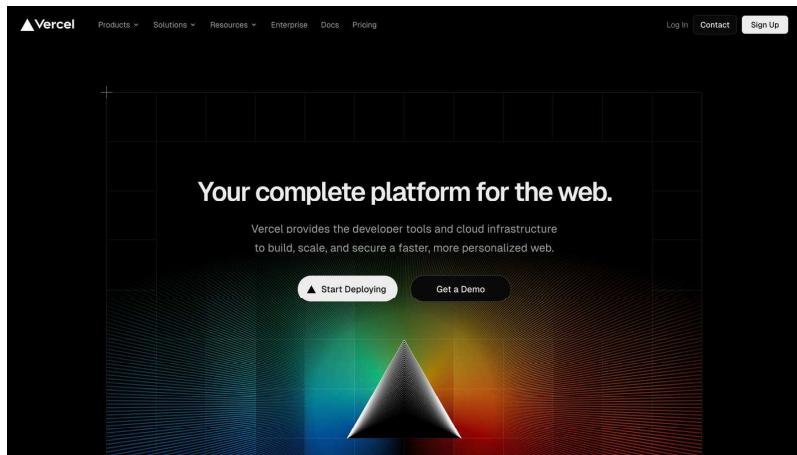
- ການ deployments ແບບບໍ່ມີການກຳນົດຄ່າ.
- ໃຊ້ງານ HTTPS ແລະ SSL ແບບອັດຕະໂນມັດ.
- Global CDN ຈັດສິ່ງຂໍ້ມູນໄວ.
- ຮອງຮັບ Serverless Functions.
- ສາມາດເບື້ງ deployments ໃນຫຼຸກງານ git push.
- ປ່ຽນແປງ domains.
- ການຈັດການຕົວແປໃນ Environment.
- Continuous Integration and Deployment (CI/CD)

ການນຳໃຊ້ໂຄງການເຂົ້າສູ່ລະບົບຈິງ ແລະ ໂຄງການສຸດທ້າຍ

ການຮອງຮັບ frameworks ຂອງ Vercel

Vercel ຕຶກຮອງຮັບຈາກຫຼາກຫຼາຍ frameworks ຍອດນີ້ຍືມ ດົວຢ່າງເຊັ່ນ:

- React (Create React App, Next.js)
- Vue.js (Nuxt.js)
- Angular
- Svelte
- And many others

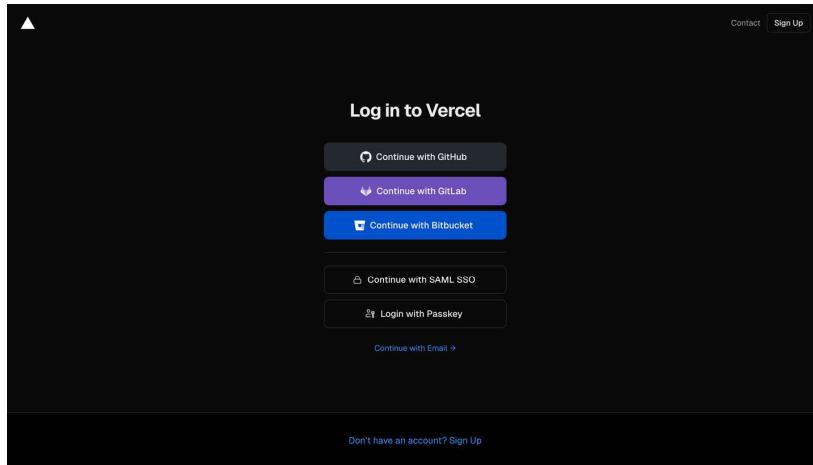


<https://vercel.com/>

ການນຳໃຊ້ໂຄງການເຂົ້າສູ່ລະບົບຈິງ ແລະ ໂຄງການສຸດທ້າຍ

ການລົງທະບຽນ/ການເຂົ້າໃຊ້ງານ Vercel

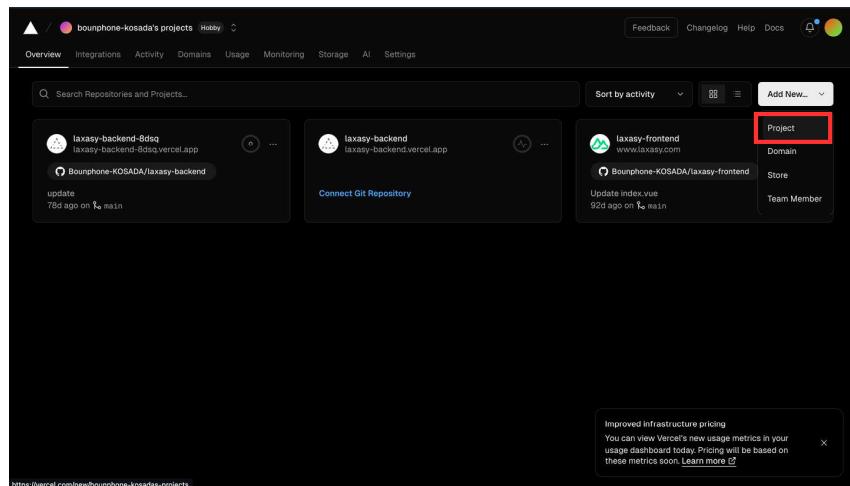
- ກັບປີເປື້ອນໜ້າ “ແນະນຳ GitHub”.
- ລົງທະບຽນ ຫຼື ເຂົ້າໃຊ້ງານ ໂດຍສາມາດໃຊ້ບັນຊີ GitHub ລົງທະບຽນໄດ້.



ການນຳໃຊ້ໂຄງການເຂົ້າສູ່ລະບົບຈິງ ແລະ ໂຄງການສຸດທ້າຍ

ໜ້າຕ່າງ Vercel

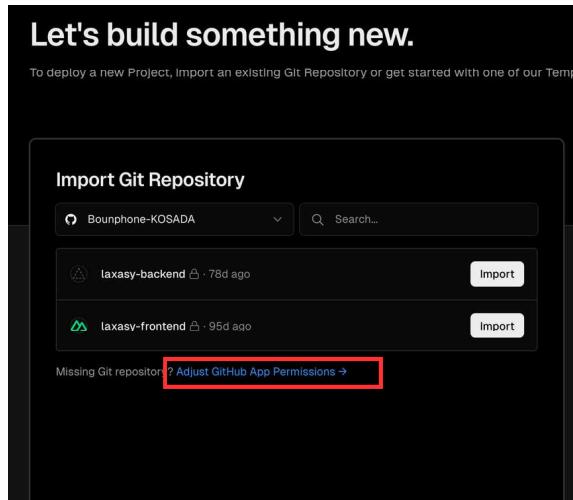
ກົດປຸ່ມ “Add New...” ແລະ ກົດ “Project”.



ການນຳໃຊ້ໂຄງການເຂົ້າສູ່ລະບົບຈິງ ແລະ ໂຄງການສຸດທ້າຍ

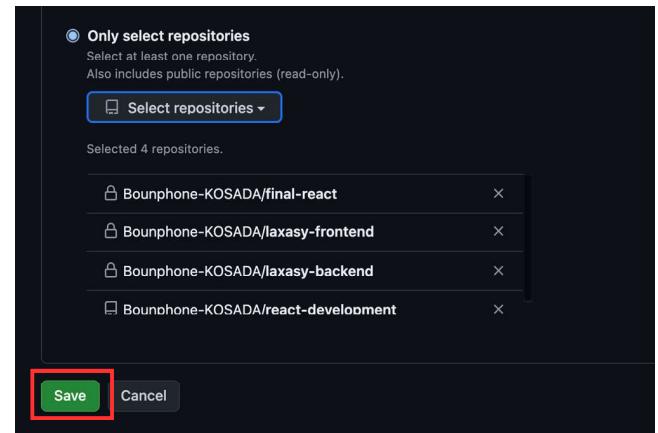
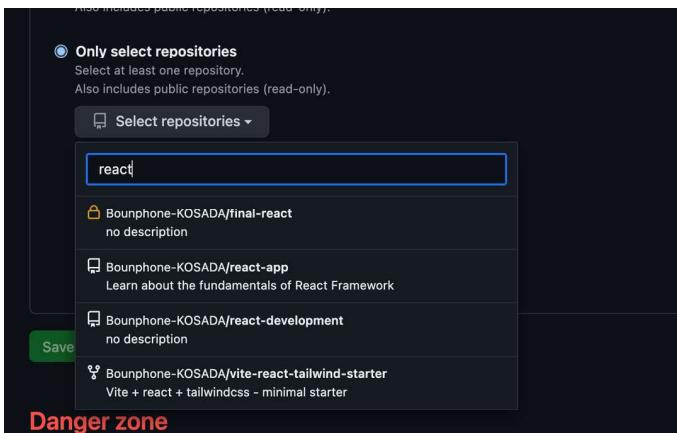
Vercel

ກຶດ “Adjust GitHub App Permissions ->”.



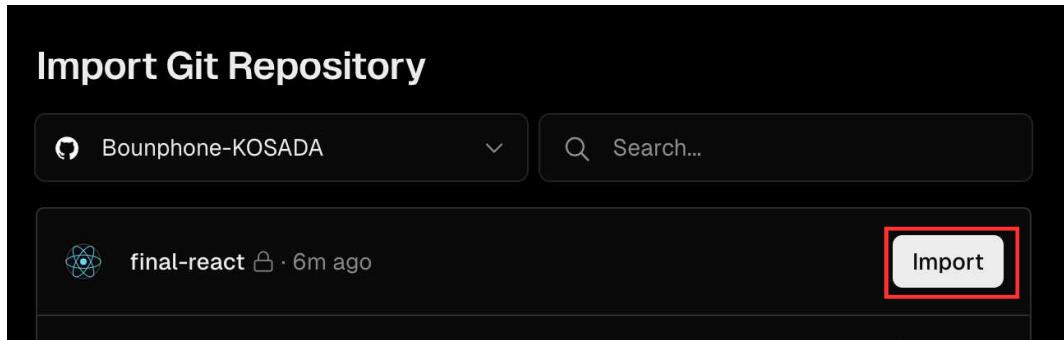
ການນຳໃຊ້ໂຄງການເຂົ້າສູ່ລະບົບຈິງ ແລະ ໂຄງການສຸດທ້າຍ

ສາມາດພິມຊື່ repository ເພື່ອຊອກຫາ repository ທີ່ຕ້ອງການໄດ້ໄວຂຶ້ນ ຫຼັງຈາກນັ້ນກົດປຸ່ມ “Save”.



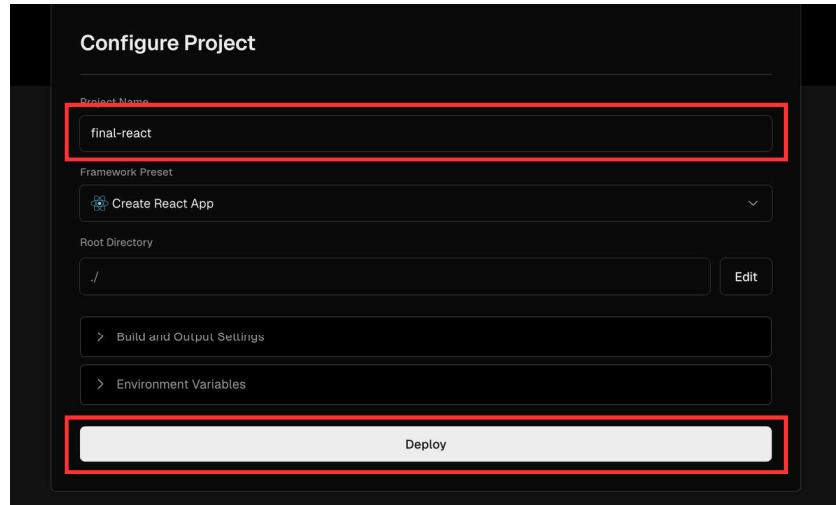
ການນຳໃຊ້ໂຄງການເຂົ້າສູ່ລະບົບຈິງ ແລະ ໂຄງການສຸດທ້າຍ

ກົດປຸ່ມ “Import” ເພື່ອນຳເອົາ repository ໄປເກັບເວີ້ນໃນ Vercel.



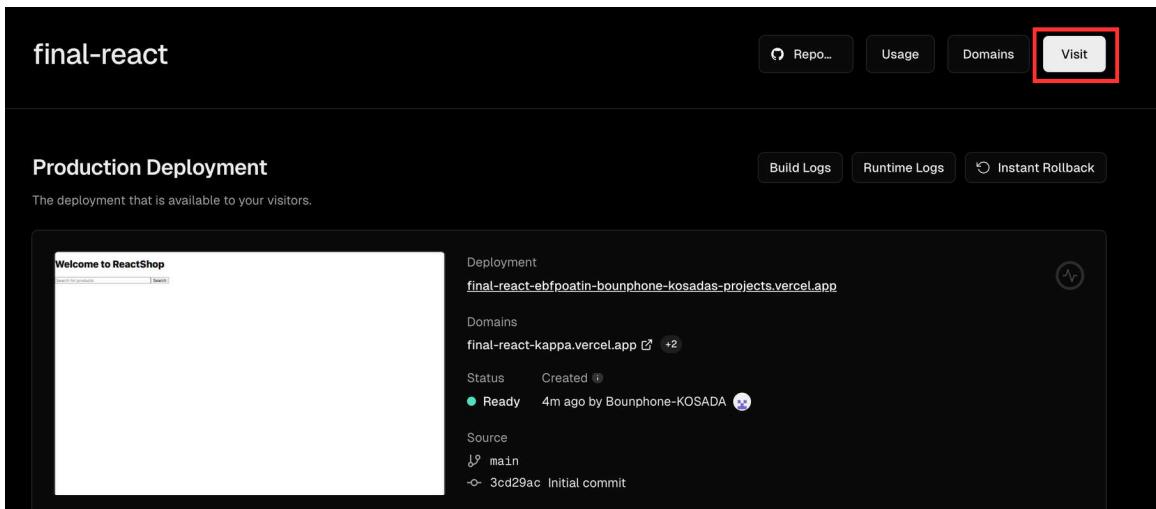
ການນຳໃຊ້ໂຄງການເຂົ້າສູ່ລະບົບຈິງ ແລະ ໂຄງການສຸດທ້າຍ

ພິມຊື່ project ໃນ “Project Name” ແລະ ກົດ “Deploy”.



ການນຳໃຊ້ໂຄງການເຂົ້າສູ່ລະບົບຈິງ ແລະ ໂຄງການສຸດທ້າຍ

ສາມາດກິດປຸ່ມ “Visit” ເພື່ອເບິ່ງເວັບໃຊ້ທີ່ອັບໂຫລດໄດ້.



ການນຳໃຊ້ໂຄງການເຂົ້າສູ່ລະບົບຈິງ ແລະ ໂຄງການສຸດທ້າຍ

ນີ້ຄືເວັບໄຊ້ຂອງບົດນີ້, ສາມາດເບິ່ງໄດ້ໂດຍການກິດໄປທີ່ລື້ງ: <https://final-react-kappa.vercel.app/>



ຂໍ້ມູນອ້າງອີງ

- <https://developer.mozilla.org/en-US/docs/Web/HTML>
- <https://developer.mozilla.org/en-US/docs/Web/CSS>
- <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide>
- <https://phoenixnap.com/kb/install-node-js-npm-on-windows>
- <https://www.geeksforgeeks.org/how-to-install-nodejs-on-macos/>
- <https://legacy.reactjs.org/docs/components-and-props.html#:~:text=Conceptually%2C%20components%20are%20like%20JavaScript,should%20appear%20on%20the%20screen.>
- <https://www.geeksforgeeks.org/reactjs-lifecycle-components/>
- <https://www.neosofttech.com/blogs/optimize-react-performance/>
- <https://www.copycat.dev/blog/react-context/>
- <https://codewithpawan.medium.com/understanding-controlled-components-in-react-a-comprehensive-guide-with-examples-931618376835>

- <https://medium.com/bb-tutorials-and-thoughts/what-is-called-lifting-state-up-in-react-785d52da940a>
- <https://vercel.com/>
- <https://github.com/>
- <https://desktop.github.com/download/>

ຄໍາຕອບບົດເຝັກທິດມື້ທີ 1

- 1.b
- 2.c
- 3.c
- 4.a
- 5.b
- 6.c
- 7.b
- 8.b
- 9.c
- 10.b

ຄໍາຕອບບົດເຝັກທິດມື້ທີ 2

- 1.b
- 2.a
- 3.b
- 4.b
- 5.b
- 6.c
- 7.b
- 8.c
- 9.b
- 10.c

ຄໍາຕອບບົດເຝັກທິດມື້ທີ 3

- 1.b
- 2.b
- 3.b
- 4.b
- 5.b
- 6.b
- 7.b
- 8.b
- 9.a
- 10.b

ຄໍາຕອບບົດເຝັກທິດມື້ທີ 4

- 1.b
- 2.b
- 3.b
- 4.b
- 5.b
- 6.b
- 7.b
- 8.b
- 9.b
- 10.b

ຄໍາຕອບບົດເຝັກທິດມື້ທີ 5

- 1.b
- 2.c
- 3.b
- 4.c
- 5.b
- 6.b
- 7.a
- 8.b
- 9.b
- 10.b

ຄໍາຕອບບົດເຝັກທິດມື້ທີ 6

- 1.b
- 2.b
- 3.b
- 4.b
- 5.b
- 6.c
- 7.a
- 8.a
- 9.b
- 10.c

ຄໍາຕອບບົດເຝັກທີ່ດີມື້ທີ 7

- 1.b
- 2.b
- 3.b
- 4.b
- 5.b
- 6.b
- 7.b
- 8.c
- 9.b
- 10.b

ຄໍາຕອບບົດເຝັກທີ່ດີມື້ທີ 8

- 1.b
- 2.c
- 3.a
- 4.a
- 5.b
- 6.b
- 7.b
- 8.c
- 9.b
- 10.b

THANK YOU!