VIBE CODING

လိုတိုရှင်း

အိမောင်

Fairway

ပုံနှိပ်မှတ်တမ်း

ထုတ်ဝေသူ	-	ဦးအိမောင် Fairway စာပေ (၀၃၃၀၇) အမှတ် 3A308၊ ရတနာနှင်းဆီအိမ်ရာ၊ ရတနာလမ်း၊ ဒဂုံဆိပ်ကမ်းမြို့နယ်၊ ရန်ကုန်တိုင်းဒေသကြီး။
ပုံနှိပ်သူ	-	ဦးအိမောင် Ebook (ဒစ်ဂျစ်တယ်စာအုပ်)
မျက်နှာဖုံးဒီဇိုင်း	-	အိမောင် (Fairway)
ကွန်ပျူတာစာစီ	-	အိမောင် (Fairway)
တန်ဖိုး	-	(၃၀,၀၀၀) ကျပ်
ဖြန့်ချီရေး	-	Fairway စာပေ (၀၉ - ၂၅၂ ၄၂၆ ၃၈၈)

မိတ်ဆက်

အားလုံးပဲ မင်္ဂလာပါ။ Vibe Coding ဆိုတာ ကနေ့ခေတ်မှာ လူတိုင်းသိထားသင့်တဲ့ ကုဒ် ရေးနည်း တမျိုးပါပဲ။ ကိုယ်တိုင်ရေးတာ မဟုတ်ဘဲ၊ ကိုယ်လိုချင်တာလေးတွေ၊ ကိုယ်ဖြစ်ချင်တာလေးတွေ ပြောပြပြီး AI ကို ရေးခိုင်းရတာပါ။

ဘာကြောင့် လူတိုင်းသိထားသင့်သလဲဆိုတော့၊ အရင်ကဆိုရင် Software တွေ App တွေ ရေးသားဖန်တီးတယ်ဆိုတာ၊ သက်ဆိုင်ရာ Coding ပညာရပ်တွေကို ကျွမ်းကျင်အောင် လေ့လာသင်ယူထားသူတွေမှသာ လုပ်နိုင်တာပါ။ ကနေ့ AI ခေတ်မှာတော့ AI ရဲ့ အကူအညီနဲ့ လူတိုင်းလုပ်လို့ရသွားပါပြီ။

ဒီတော့ ဒီအလုပ်နဲ့ ကျွမ်းကျင်သူတစ်ဦးအဖြစ် ရပ်တည်ချင်သူ၊ ဝင်ငွေရှာဖွေချင်သူ မဟုတ်ရင်တောင်မှ၊ ကိုယ်တိုင်သုံးဖို့၊ ကိုယ့်အလုပ်မှာသုံးဖို့၊ App လေးတွေ ကိုယ့်ဘာသာ

လုပ်တတ်အောင် လေ့လာထားတယ်ဆိုတာ လုပ်သင့်တဲ့ကိစ္စ ဖြစ်သွားပါပြီ။ ဒါကြောင့် လူတိုင်း သိထားသင့်တဲ့ အကြောင်းအရာ လို့ ပြောတာပါ။

“Coding နားမလည်ဘဲ Vibe Coding တကယ်လုပ်လို့ရလို့လား”

“**မရပါဘူး**” လို့ ပြောရင် မမှန်သလို “**ရပါတယ်**” လို့ ပြောရင်လည်း မပြည့်စုံပါဘူး။

“Coding နားမလည်လည်း Vibe Coding လုပ်လို့ရတယ်၊ အသုံးဝင်တဲ့ App တွေ လုပ်လို့ရတယ်။ Coding နားလည်ရင်တော့ အများကြီး ပိုကောင်းတာတွေ လုပ်နိုင်လိမ့်မယ်။”

- လို့ အပြည့်အစုံ ဖြေရမှာဖြစ်ပါတယ်။

ဥပမာ - ကိုယ့်လုပ်ငန်းအတွက် ဝက်ဘ်ဆိုက်လေး လုပ်ချင်တယ်။ နေ့စဉ် ဝင်ငွေထွက်ငွေ မှတ်တမ်း App လေး လုပ်ချင်တယ်။ ပစ္စည်းရောင်းလို့ရတဲ့ Pre-Order App လေးလုပ်ချင်တယ်။ စသည်ဖြင့် တစ်ကိုယ်ရေသုံးနဲ့ အသေးစား အသုံးဝင်တာလေးတွေ အများကြီးကို Coding နားမလည်လည်း Vibe Coding နဲ့ လုပ်လို့ရပါတယ်။

ဒါက စာရေးသူရဲ့ ခေါင်းထဲမှာ ပေါ်လာတာလေးတချို့ကို ချရေးလိုက်တာပါ။ လူတိုင်းလူတိုင်း အားလုံးမှာ ကိုယ်စီ လိုအပ်ချက်လေးတွေ၊ အိုင်ဒီယာလေးတွေ ရှိကြပါလိမ့်မယ်။ အဲ့ဒီလိုအပ်ချက်လေးတွေကို Vibe Coding နဲ့ ဖြေရှင်းလို့ရပါတယ်။ အဲ့ဒီ အိုင်ဒီယာလေးတွေကို Vibe Coding နဲ့ အကောင်အထည် ဖော်လို့ရပါတယ်။

ဒီစာအုပ်ရဲ့ ရည်ရွယ်ချက်က စာဖတ်သူပေါ်မူတည်ပါတယ်။

၁။ စာဖတ်သူက Vibe Coding ဆိုတာကို လုံးဝ မသိသေးတဲ့သူ ဖြစ်တယ်။ စမ်းမကြည့်ဖူးသေးတဲ့သူ ဖြစ်တယ်ဆိုရင် ဒီစာအုပ်ဖတ်ပြီးသွားတဲ့အခါ အဲ့ဒီ Vibe Coding နည်းကို သုံးပြီး ကိုယ်တိုင် App လေးတွေ ဖန်တီးနိုင်သွားမှာပါ။

၂။ စာဖတ်သူက Vibe Coding လုပ်ကြည့်ဖူးတယ်။ ရသင့်သလောက်လည်း ရတယ်။ ဒါပေမဲ့ တကယ်သုံးလို့ရတဲ့အဆင့်ထိ မရောက်ဘူးဖြစ်နေတယ် ဆိုရင် ဒီစာအုပ် ဖတ်ပြီးသွားတဲ့အခါ တကယ်သုံးလို့ရတဲ့ အဆင့်ထိ လုပ်နိုင်သွားမှာပါ။

၃။ စာဖတ်သူက Vibe Coding ကို တော်တော် ရနေပြီ။ ဒါပေမဲ့ Coding ပိုင်းနည်းပညာတွေကို မရလို့ အဆင့်တစ်ခုမှာ ရှေ့ဆက်သွားလို့မရ ဖြစ်နေတယ် ဆိုရင် ဒီစာအုပ် ဖတ်ပြီးသွားတဲ့အခါ ဘယ်လို နည်းပညာတွေကို ဆက်လက်လေ့လာရမလဲ သိသွားမှာပါ။

၄။ စာဖတ်သူက ကျွမ်းကျင် Programmer တစ်ဦး ဆိုရင်တော့ ဒီစာအုပ်ဖတ်ပြီးသွားတဲ့အခါ စာအုပ်မှာဖော်ပြထားတာတွေနဲ့ ကိုယ့်ရဲ့ကျွမ်းကျင်မှုတွေ ပေါင်းစပ်ပြီး AI ခေတ်ရဲ့ အကျိုးရလဒ်ကို အပြည့်အဝအသုံးချနိုင်သူ Programmer တစ်ဦး ဖြစ်သွားမှာပါ။

ဒီစာအုပ်မှာဖော်ပြမယ့် အကြောင်းအရာတွေကို Coding တွေ လုံးဝမသိတဲ့ လူတစ်ဦး ဖတ်လို့ရအောင် ရေးသားဖော်ပြမှာပါ။ ဒါပေမဲ့ စာဖတ်သူရဲ့ လက်ရှိ နည်းပညာဗဟုသုတ အခြေအနေက ဘယ်အဆင့်ပဲ ဖြစ်ဖြစ် ဒီစာအုပ်ပါ ဖော်ပြချက်တွေက တစ်နည်းမဟုတ် တစ်နည်း အသုံးဝင်လိမ့်မယ်လို့ ယုံကြည်ပါတယ်။

စလိုက်ကြရအောင်။

အခန်း (၁) - ကွန်ပျူတာဆော့ဖ်ဝဲများ၏ အလုပ်လုပ်ပုံ

ကွန်ပျူတာဆိုတာ အိမ်က စားပွဲပေါ်မှာတင်ထားတဲ့ ကွန်ပျူတာမှသာ ကွန်ပျူတာ မဟုတ်ပါဘူး။ အားလုံးရဲ့ လက်ထဲမှာ ရှိနေတယ် ဖုန်းတွေဟာ ကွန်ပျူတာတစျိုးဖြစ်ပြီး အင်တာနက် ဆာဗာ တွေဟာလည်း ကွန်ပျူတာတွေပါပဲ။

ဒါကြောင့် ဒီနေရာမှာ ကွန်ပျူတာဆော့ဖ်ဝဲဆိုတာ အိမ်ကကွန်ပျူတာမှာ ထည့်သုံးရတဲ့ ဆော့ဖ်ဝဲတွေလည်း ပါသလို၊ ဖုန်းထဲက App တွေလည်း ပါပါတယ်။ URL လိပ်စာ ရိုက်ထည့်ပြီးတော့ပဲဖြစ်ဖြစ်၊ လင့်နှိပ်ပြီးတော့ပဲဖြစ်ဖြစ် သုံးရတဲ့ ဝက်ဘ်ဆိုက်တွေလည်း ပါပါတယ်။

အားလုံးက ကွန်ပျူတာဆော့ဖ်ဝဲတွေချည်းပါပဲ။

ဆော့ဖ်ဝဲအမျိုးအစားမတူရင် ဖန်တီးပုံ အလုပ်လုပ်ပုံတွေလည်း မတူကြပါဘူး။ ဒါပေမဲ့ Vibe Coding လုပ်လိုသူဟာ အဲ့ဒီလို အသေးစိတ်တွေထက်၊ သဘောတရား ရင်းမြစ်ကို အရင်ဆုံး ရှာဖွေလေ့လာရပါတယ်။

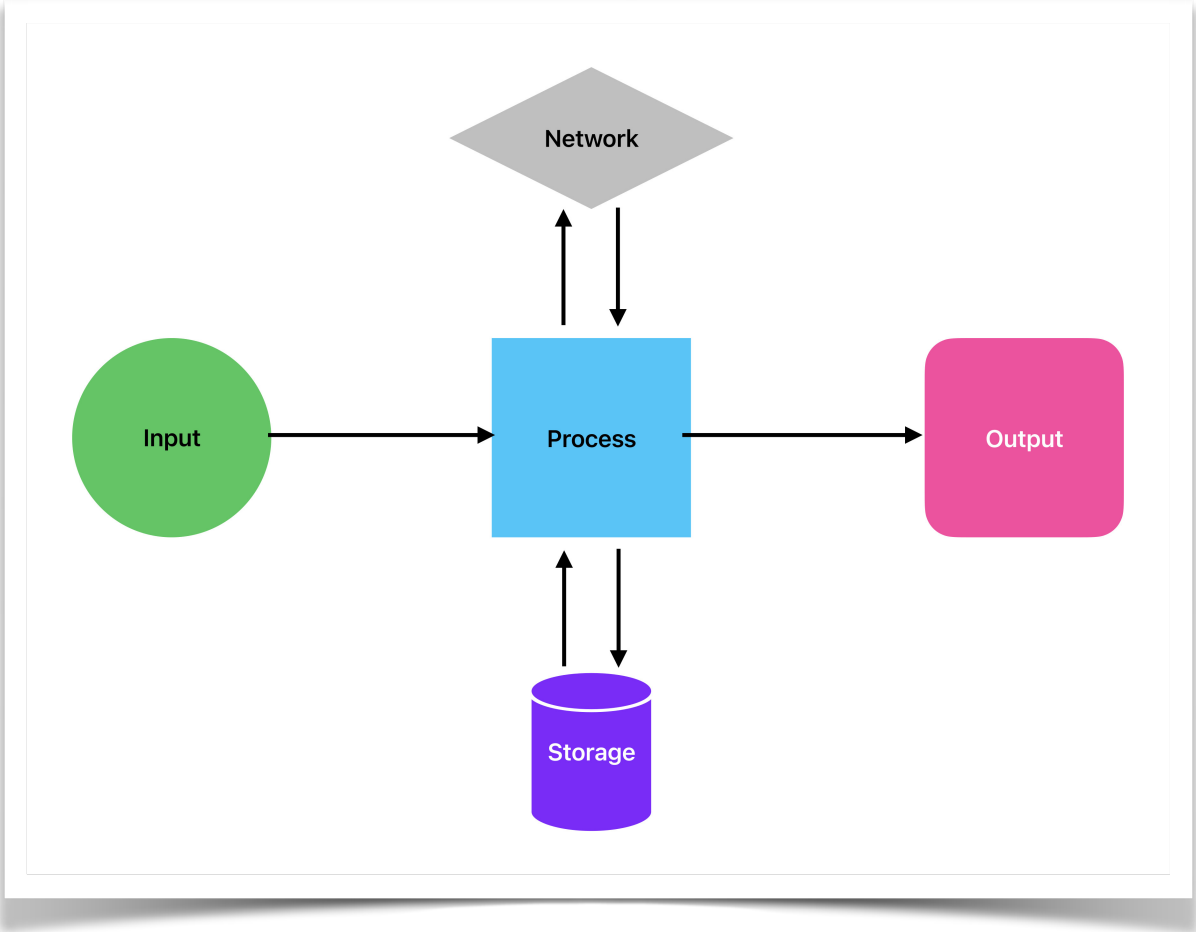
ဆော့ဖ်ဝဲအမျိုးအစားတွေ ဘယ်လောက်ပဲ ကွဲပြားပါစေ၊ အရင်းစစ်လိုက်တဲ့အခါ မည်သည့်ဆော့ဖ်ဝဲမဆို ဒီအစိတ်အပိုင်း (၅) ခုနဲ့ စုဖွဲ့ထားကြတာချည်းပါပဲ။

1. Input
2. Process
3. Output
4. Storage
5. Network

Input

Input အကြောင်းမပြောခင် အရင်ဆုံး အသုံးပြုသူ User က ဘယ်သူလဲ ဆိုတာကို အရင်ပြောဖို့လိုပါတယ်။ ကိုယ်ကိုယ်တိုင်လား။တခြားလူပေါင်းစုံလား။ အသုံးပြုသူက လူမဟုတ်ဘဲ တခြားကွန်ပျူတာဆော့ဖ်ဝဲတွေလား။ ဒါလည်း ဖြစ်နိုင်ပါတယ်။ အသုံးပြုသူက AI Agent လား။ ဒါလည်း ဖြစ်နိုင်တာပါပဲ။

Input ဆိုတာ အသုံးပြုသူက ပေးပို့လာတဲ့ အချက်အလက်တွေပါ။ ဖောင်မှာ ရေးဖြည့်ပေးပို့တာ ဖြစ်နိုင်တယ်။ Button တွေ လင့်တွေကို နှိပ်ပြီး ပေးပို့တာ ဖြစ်နိုင်တယ်။ အသံနဲ့



ပေးပို့တာ ဖြစ်နိုင်တယ်။ QR ကုဒ်ကို Scan ဖတ်ပြီး ပေးပို့တာ ဖြစ်နိုင်တယ်။ ဖိုင်အနေနဲ့ ပေးပို့တာ ဖြစ်နိုင်ပါတယ်။

ဒါကြောင့် သုံးမယ့်သူက ဘယ်သူလဲ၊ ဘာတွေပေးပို့မှာလဲဆိုတာ Input ဆိုတဲ့ ကဏ္ဍမှာ စဉ်းစားအဖြေရှာ ကြိုတင်မေးကြည့်ရမယ့် မေးခွန်းတွေဖြစ်ပါတယ်။

ဒါတွေကို ကိုယ်တိုင်စဉ်းစားလို့ရသလို AI နဲ့လည်း တိုင်ပင်လို့ရပါတယ်။ ဥပမာ - လုပ်ချင်တဲ့ ဆော့ဖ်ဝဲအမျိုးအစားကို ပြောပြပြီး အသုံးပြုသူက ဘယ်လိုပုံစံဖြစ်နိုင်လဲ၊ Input တွေက ဘယ်လိုပုံစံဖြစ်နိုင်လဲ မေးကြည့်၊ တိုင်ပင်ကြည့်လို့ရပါတယ်။

အရာရာ ကိုယ်သိစရာမလိုပါဘူး။ ဘာကိုသိသင့်တယ်၊ ဘယ်လိုမေးခွန်းတွေ ထုတ်ရမယ် ဆိုတာကို သိရင်ရပါပြီ။

Process

အသုံးပြုသူပေးပို့တဲ့ Input တွေနဲ့ ဘာအလုပ်တွေ လုပ်ပေးမှာလဲ ဆိုတာဟာ Process ဖြစ်ပါတယ်။

အသုံးစရိတ်ထည့်ပေးလိုက်ရင် တစ်လစာ ဝင်ငွေထဲက အဲ့ဒီအသုံးစရိတ်ကို နှုတ်ပေးမှာလား။ တစ်နှစ်စာထဲကရော နှုတ်ဦးမှာလား။ နှုတ်လိုက်လို့ အနှုတ်ပြနေရင် ဘာလုပ်ရမလဲ။ စသည်ဖြင့်။

Pre-Order မှာဖို့ ပေးပို့တဲ့ အချက်အလက်တွေ မှန်မမှန်စစ်ပေးမှာလား။ အရစ်ကျတွေ လက်ခံမှာလား။ ငွေ Amount မပြည့်ရင် ဘာလုပ်မှာလဲ။ ပြည့်သွားရင် ဘာလုပ်မှာလဲ။ ကျော်သွားရင်ရော ဘာလုပ်မှာလဲ။ ဘာနဲ့ပေးရမှာလဲ။ ဘယ်လိုစစ်မှာလဲ။

Comment တစ်ခုဖျက်တာဆိုရင် အဲ့ဒီ Comment ကို ဖျက်ခွင့်ရှိတဲ့သူ ဟုတ်မဟုတ်စစ်။ ဖျက်ခွင့်မရှိရင် ဘာလုပ်မှာလဲ။ ဖျက်ခွင့်ရှိရင် ဘာဆက်လုပ်မှာလဲ။

စသည်ဖြင့် အလုပ်လုပ်မယ့် ကုဒ်အသေးစိတ်ကို သိစရာမလိုပေမဲ့ ဘာဖြစ်ချင်တာလဲ ဘာလုပ်ချင်တာလဲဆိုတာကိုတော့ အသေးစိတ်သိလေ ကောင်းလေပါပဲ။

အသေးစိတ်သိမှပဲ ရမှာမျိုးတော့ မဟုတ်ပါဘူး။ အကြမ်းဖျဉ်းအိုင်ဒီယာလောက် ရှိရင် အသေးစိတ်ကို AI နဲ့ပဲ ဆက်လက်တိုင်ပင်ကြည့်လို့လည်း ရပါတယ်။

Vibe Coding တွေမလုပ်ခင်၊ App တွေ မဖန်တီးခင် ဒါလေးတွေကို အရင်ဆုံး ကိုယ်တိုင် စဉ်းစား အဖြေထုတ်ကြည့်ရမှာ ဖြစ်ပါတယ်။ နောက်ပိုင်း အကျင့်ရသွားရင် အမြဲတမ်း ဒီအဆင့်တွေကို တစ်ဆင့်ချင်း ကျော်ဖြတ်နေစရာ မလိုတော့ပါဘူး။ လုပ်နေကျမဟုတ်ရင် အစပိုင်းမှာ ဒီအဆင့်တွေကနေ အရင်စသင့်ပါတယ်။

Output

လုပ်စရာရှိတဲ့ Process တွေ လုပ်ပြီးနောက် ရလဒ် Result ကို အသုံးပြုသူ User ထံ ပြန်ပေးမယ့် အချက်အလက်တွေ၊ ပြန်ပြမယ့်အရာတွေက Output ပါပဲ။

Screen မှာ ပြမှာလား။ စာနဲ့ ပြမှာလား။ စာရင်းဇယားနဲ့ ပြမှာလား။ ရုပ်ပုံနဲ့ ပြမှာလား။ ဖိုင်နဲ့ပြန်ပေးမှာလား။ Message ပို့ပေးမှာလား။ Notification ပို့ပေးမှာလား။

ဖြစ်နိုင်တာတွေ အများကြီး ရှိပါတယ်။ အများစုကတော့ Screen မှာပဲ Box လေးတွေ၊ Block လေးတွေ၊ Color တွေ၊ Text တွေ၊ List တွေသုံးပြီး User Interface (UI) အနေနဲ့ ပြန်ပြတာ ဖြစ်နိုင်ပါတယ်။

အဲ့ဒီလို ပြန်ပြတဲ့အချိန်မှာ နောက်တစ်ဆင့် ဆက်သွားဖို့၊ နောက်အလုပ်တွေ ဆက်လုပ်ဖို့ Input ပေးလို့ရတဲ့ Button တွေ Text Box တွေ Link တွေလည်း ပြန်ပါကောင်း ပါလိမ့်မယ်။

ဥပမာ - Pre-Order ပို့လိုက်တဲ့အခါ၊ အချက်အလက်တွေ မှားနေရင် မှားနေကြောင်းပြန်ပြတာ ဖြစ်နိုင်ပါတယ်။ အချက်အလက်ပြည့်စုံရင် Receipt ကို ပြန်ပြတာ ဖြစ်နိုင်ပါတယ်။

ဒီ Process၊ ဒီ Flow ကို ကိုယ့်ခေါင်းထဲမှာ အရင်ဆုံးရှင်းထားဖို့က တော်တော် အရေးကြီးပါတယ်။ Vibe Coding လုပ်ကြည့်တဲ့အခါ ရလဒ်ကောင်းတွေထွက်မထွက်ဆိုတာ ကိုယ်လိုချင်တာဘာလဲ ကိုယ့်ဘာသာ ရှင်းမရှင်း ဆိုတဲ့ပေါ်မှာ တော်တော်ကြီးမူတည်ပါတယ်။

Coding တွေဘာတွေ ရရဲ့သားနဲ့ Vibe Coding ကျတော့ အဆင်မပြေဘူး ဆိုတဲ့သူတွေ ရှိနိုင်ပါတယ်။ ကိုယ်ဘာလိုချင်လဲ ကိုယ့်ဘာသာ မရှင်းလို့ ဖြစ်နေတာ များပါတယ်။

Storage

အသုံးပြုသူ User တွေရဲ့ အချက်အလက်တွေနဲ့ Process လုပ် ရရှိလာတဲ့ ရလဒ်တွေကို တစ်နေရာရာမှာတော့ သိမ်းထားဖို့ လိုပါလိမ့်မယ်။ ဒါ Storage ပါပဲ။

အသုံးပြုသူရဲ့ Device (ဖုန်း၊ ကွန်ပျူတာ) ထဲမှာ သိမ်းမှာလား။ အင်တာနက်ဆာဗာတခုခုနဲ့ သိမ်းမှာလား။ ဖိုင်အနေနဲ့ သိမ်းမှာလား။ Database နည်းပညာတွေနဲ့ သိမ်းမှာလား။

တချို့အချက်အလက်တွေက ဖုန်းတို့ ကွန်ပျူတာတို့လို Device ထဲမှာတင် သိမ်းလို့ရပါတယ်။ အထူးသဖြင့် တစ်ဦးတည်း သုံးမယ့် App တွေပါ။ လူအများကြီး ဝိုင်းသုံးလို့ ရချင်တယ်ဆိုရင်တော့ Device ထဲမှာ သိမ်းထားရုံနဲ့ မလုံလောက်တော့ပါဘူး။

Database တွေ Server တွေ လိုလာပါတယ်။ အဲ့ဒါတွေ မသိဘူး။ အဆင်ပြေပါ့မလား။

ဒါဟာ အဓိကပြဿနာပါပဲ။ AI က ကုဒ်တွေကို အလွယ်တကူ ရေးပေးနိုင်ပေမဲ့ ကိုယ့်အကူအညီမပါဘဲ Database တွေ Server တွေကို စီမံနိုင်မှာ မဟုတ်ပါဘူး။ စီမံတာက စီမံနိုင်ပါတယ်။ သူ့ချည်းမရဘဲ ကိုယ့်အကူအညီ လိုနေတာပါ။

ကိုယ့်ဘက်က အကောင့်တွေ ဆောက်ပေးရမယ်။ အကောင့်ဆောက်ပြီးနောက် ရလာတဲ့ အချက်အလက်တွေ AI ကို ပေးရမယ်။ ဒါဆိုရင် AI က Database တွေ Server တွေထိ စီမံနိုင်သွားပါပြီ။

ထုံးစံအတိုင်း လုပ်နည်းအသေးစိတ်ထက် အဲ့ဒီလို လိုအပ်ချက်ရှိတယ်၊ ကိုယ့်အကူအညီနဲ့ ဖြေရှင်းလို့ရတယ်ဆိုတာကို သိဖို့က ပိုအရေးကြီးပါတယ်။ ဒီလိုသိပြီဆိုရင် AI ကိုပဲ ဘယ်နည်းပညာ သုံးရမလဲ၊ ငါဘာလုပ်ပေးရမလဲ တိုင်ပင်လို့ ရသွားပါပြီ။

လက်တွေ့နမူနာတွေ နောက်ပိုင်းမှာ ထည့်သွင်းဖော်ပြပေးမှာပါ။

Network

တစ်ဦးတည်းသုံးမယ့် App တွေကို Standalone App လို့ ခေါ်ပါတယ်။ တချို့ App တွေက တစ်ဦးတည်းသုံးမယ်၊ အင်တာနက်တွေဘာတွေလည်း မလိုဘူး ဆိုတာမျိုး ဖြစ်နိုင်ပါတယ်။ ဒီလို App မျိုးကို Offline App လို့လည်း ခေါ်နိုင်ပါတယ်။

တစ်ဦးတည်း မဟုတ်တော့ဘူး၊ အများဝိုင်းဝန်းအသုံးပြုမှာ ဆိုရင်တော့ Network ခေါ် အပြန်အလှန် ချိတ်ဆက် ဆက်သွယ်မှုတွေ လိုလာပါပြီ။

အများအားဖြင့် အသုံးများတဲ့ ပုံစံက၊ ဗဟိုစနစ်တစ်ခုကို Device တွေအများကြီးက လှမ်း ချိတ်ပြီး သုံးတဲ့ စနစ် ဖြစ်ပါတယ်။ ဗဟိုစနစ်မှာ စောစောက ပြောခဲ့တဲ့ Database ပိုင်း လုပ်ဆောင်ချက်တွေ ရှိနေပါလိမ့်မယ်။ Backend လို့ ခေါ်ကြပါတယ်။

ချိတ်ဆက်မယ့် အသုံးပြုသူ User တွေရဲ့ Device ထဲမှာ ကိုယ့်ရဲ့ App က ရှိနေမှာဖြစ်ပါတယ်။ Frontend လို့ ခေါ်ပါတယ်။

ပြီးတဲ့အခါ ငွေပေးချေမှုတွေ၊ SMS ပေးပို့မှုတွေ၊ ဒေါ်လာဈေး၊ ငွေဈေး၊ ရာသီဥတုလို အချက်အလက်တွေကို ဝန်ဆောင်မှုအနေနဲ့ ပေးနိုင်တဲ့ Service တွေ အများကြီး ရှိကြပါတယ်။ အရာရာ ကိုယ်တိုင် လုပ်စရာမလိုဘဲ သက်ဆိုင်ရာ Service တွေနဲ့ ချိတ်ဆက် အသုံးပြုလို့ရပါတယ်။ အဲ့ဒီလို ချိတ်ဆက် အသုံးပြုဖို့ Network လိုလာတာပါ။

ခက်လာပါပြီ။ ရှုပ်လာပါပြီ။ ဒါပေမဲ့ ပြဿနာကြီး မဟုတ်ပါဘူး။

ထုံးစံအတိုင်း ဒါတွေဒီလို ရှိတယ်ဆိုတာကို သိထားရင်ကို တော်တော် ဟုတ်နေပါပြီ။ ဘယ်လို Service တွေကို ငါတို့ ချိတ်သုံးကြမလဲ၊ ဘယ်လိုကိစ္စတွေ ငါတို့ ကိုယ့်ဘာသာ လုပ်ကြမလဲ၊ ငါ့ဘက်က ဘာလုပ်ပေးရမလဲဆိုတာ AI နဲ့ တိုင်ပင်လိုက်ရင် အသေးစိတ် ပြောပြပါလိမ့်မယ်။

ဆော့ဖ်ဝဲတွေရဲ့ ရင်းမြစ် (၅) ချက်ကို သိသွားပြီဆိုတော့၊ နောက်တစ်ဆင့်မှာ အဲ့ဒီ အကြောင်းအရာ (၅) ခုကို ချိတ်ဆက်ရပါမယ်။ ဒီနေရာမှာ အသုံးဝင်လာတာက System Thinking လို့ခေါ်တဲ့ သဘောတရားတစ်ခုပါ။

System Thinking

တကယ့် System Thinking ဆိုတာ Feedback Loop တို့ Emergence တို့လို သဘော တရားတွေနဲ့ တော်တော်ကျယ်ပြန့်တဲ့ Engineering Practice တစ်ခုပါ။ ဒီနေရာမှာတော့ System Thinking ဆိုတာ စောစောက ပြောခဲ့တဲ့ အစိတ်အပိုင်းလေးတွေ တစ်ခုနဲ့တစ်ခု ဘယ်လိုချိတ်ဆက် စီးဆင်းအလုပ်လုပ်သလဲ ဆိုတာကို တွေးခေါ်မြင်ကြည့်နိုင်စွမ်းလို့ အလွယ်မှတ်ယူနိုင်ပါတယ်။ AI ခေတ်မှာ အရေးအကြီးဆုံး အရည်အချင်းတစ်ခုပါပဲ။

ကိုယ်ဖန်တီးလိုတဲ့ App ရဲ့ အသေးစိတ်ဖော်ပြပုံတွေကို ခဏမေ့ထားပြီး ဘယ်ကစမှာလဲ၊ ဘယ်ကိုသွားမှာလဲ၊ ပြီးရင် ဘာဆက်ဖြစ်မှာလဲ၊ နောက်ဆုံးရလဒ်က ဘာလဲဆိုတဲ့ ချိတ်ဆက်စီးဆင်းမှုကို စဉ်းစားကြည့်ပါ။

အများအားဖြင့် လုပ်ငန်းစဉ်တစ်ခုဟာ Input ကနေစပါတယ်။ ဖြစ်နိုင်ခြေ အမျိုးမျိုး ရှိပါတယ်။

- **Input → Process → Output**

အသုံးပြုသူပေးတဲ့ Input အချက်အလက်တွေကို လက်ခံပြီးနောက် လုပ်စရာရှိတဲ့ Process တွေလုပ်ပြီး၊ ရလဒ် Output ပြန်ပြမှာလား။

- **Input → Network → Process → Output**

Process အလုပ်တွေ မလုပ်ခင် Network နဲ့ External Service ကို လှမ်းချိတ်ဆက်ပြီးမှ ဆက်အလုပ်လုပ်မှာလား။

- **Input → Process → Storage → Output**

Process အလုပ်လုပ်ပြီးနောက် ရရှိလာတဲ့ အချက်အလက်တွေကို Storage မှာ အရင် သိမ်းဆည်းဦးမှာလား။ စသည်ဖြင့် ကိုယ်ဖန်တီးလိုတဲ့ App ရဲ့ စီးဆင်းမှု Flow ကို မြင်ကြည့်ဖို့လိုပါတယ်။

ကောင်းကောင်းမမြင်ဘူး ဖြစ်နေရင်လည်း AI နဲ့ တိုင်ပင်လို့ရပါတယ်။ သူ့ကို Diagram တွေ Flow Chart တွေ ဆွဲခိုင်းလို့ရပါတယ်။ မြင်သွားပြီဆိုရင် လွဲချော်မှုတခုခု ရှိလာတဲ့အခါ ဘယ်အဆင့်မှာ၊ ဘယ် Flow မှာ လွဲနေတာလဲ ဆိုတာကို ကိုယ်တိုင်မှန်းဆသိရှိပြီ AI ကို ပိုတိတိကျကျ လမ်းညွှန်နိုင်သွားမှာပါ။

ဒီနည်းနဲ့ဆိုရင်၊ စမ်းကြည့်တာ ရမလိုလို ရမလိုလိုနဲ့ အပြီးသတ်နားကျမှ အဆင်မပြေဖြစ်နေတယ်၊ ဖြေရှင်းလို့မရတော့ဘူး ဖြစ်နေတယ်ဆိုတာတွေ ပြေလည်သွားပါလိမ့်မယ်။

နောက်အခန်းတွေ ဆက်မဖတ်သေးဘဲ၊ ဒီနေရာမှာ ခဏရပ်ထားပြီး၊ ဒီဗဟုသုတနဲ့၊ ကိုယ့်ဘက်က လိုချင်တာလေးတွေ၊ ဖြစ်စေချင်တာလေးတွေ ပြောပြီး သိချင်တာလေးတွေ AI နဲ့ ပြန်တိုင်ပင်လိုက်ရင် တော်တော်အဆင်ပြေတဲ့ App တွေ စလုပ်လို့ရနေပါပြီ။

အခန်း (၂) - AI အလုပ်လုပ်ပုံ

ကွန်ပျူတာစနစ်တွေဟာ **အယ်ဂေါရစ်သမ်** ခေါ် ကြိုတင်သတ်မှတ်ထားတဲ့ တိကျတဲ့ စည်းမျဉ်းတွေကို ကောင်းကောင်း လိုက်နာနိုင်ကြပါတယ်။ ဥပမာ -

1. ကိန်းဂဏန်းတွေ အများကြီးရှိတဲ့ထဲက၊
2. တစ်ကြိမ်မှာ နှစ်ခုကိုယူပါ။
3. အကြီးအသေး နှိုင်းယှဉ်ပါ။
4. ကြီးရင် ညာမှာထား၊ သေးရင် ဘယ်မှာထားပါ။
5. အားလုံးငယ်စဉ်ကြီးလိုက် ညီသွားတဲ့ထိ ထပ်ခါထပ်ခါပြန်လုပ်ပါ။

- လို့ ပြောထားလိုက်ရင်၊ များပြားတဲ့ ကိန်းဂဏန်းတွေကို ငယ်စဉ်ကြီးလိုက် ညီအောင် စီပေးနိုင်သွားပါလိမ့်မယ်။ ဒီလို တိကျတဲ့ အဆင့်လိုက်လုပ်ငန်းစဉ် အယ်ဂေါရစ်သမ်တွေနဲ့

ကွန်ပျူတာတွေက အလုပ်တွေကို လူထက်ပိုပြီး များများနဲ့မြန်မြန် လုပ်နိုင်ကြပါတယ်။ တိတိကျကျလည်း လုပ်နိုင်ကြပါတယ်။

ဒါပေမဲ့ ဒီလိုအတိအကျ ကြိုတင်လုပ်ငန်းစဉ် သတ်မှတ်ပေးထားလို့ မရနိုင်တဲ့ ပြဿနာတွေလည်း ရှိနေပါတယ်။

ဥပမာ - “**နေကောင်းလား**” လို့မေးရင် “**ကောင်းပါတယ်**” လို့ ပြန်ဖြေပါလို့ တိတိကျကျ သတ်မှတ်ထားတယ် ဆိုကြပါစို့၊ “**နေကောင်းရဲ့လား**” လို့ မေးလာခဲ့ရင် ပြန်ဖြေနိုင်တော့မှာ မဟုတ်ပါဘူး။ ကြိုတင်သတ်မှတ်ထားတဲ့ စည်းမျဉ်းနဲ့ မတူတော့လို့ပါ။

ဒီတော့ ဘယ်လိုလုပ်ကြမလဲ။ စည်းမျဉ်းထဲမှာ -

- “**နေကောင်းလား**”
- “**နေကောင်းရဲ့လား**”
- “**နေကောင်းပါသလား**”

- စသည်ဖြင့် အကုန်စုံအောင် ထည့်ထားမှာလား။ အဲ့ဒီလို ထည့်ထားလိုက်သည့်တိုင်အောင် “**နေကောင်း ရဲ့လား**” ဆိုရင် မဖြေနိုင်တော့ပြန်ဘူးဆိုတာ ဖြစ်နိုင်ပါတယ်။ အလယ်မှာ Space တစ်ခုက ပါသွားပြန်ပြီ။

စည်းမျဉ်းထဲမှာ Space တွေ အကုန်ဖယ်ထုတ်ဖို့ ထပ်ထည့်မှာလား။ ဒါဆိုရင်လည်း “**မိသားစုတွေကျန်းမာကြရဲ့လား**” ဆိုရင် မသိတော့ပြန်ဘူး။ ကြိုတင်သတ်မှတ်ထားတဲ့

စည်းမျဉ်းတွေနဲ့ ဘယ်လိုမှ ဖြေရှင်းလို့ မရနိုင်တော့ပါဘူး။ ဒီလိုပြဿနာတွေ အများကြီး ရှိနေပါတယ်။

AI Models

ဒီလိုအခြေအနေမျိုးမှာ ဖြစ်နိုင်ခြေရှိတဲ့ နှုတ်ဆက်စကားတွေကို ရှာဖွေစုဆောင်း၊ ပါတဲ့ စာလုံးတွေကို ခွဲထုတ်၊ အကုန်လုံးကို ထုခြေ (ကိန်းဂဏန်းတွေ ဖြစ်အောင်ပြောင်း)၊ ရလာတဲ့ ပုံသဏ္ဌာန် အထူအပါး၊ အနီးအဝေး၊ အစိတ်အကျဲကို အလွှာလေးတွေခွဲပြီး မှတ်ထားလိုက်တယ် ဆိုပါစို့။ စကားလုံးတွေကို ပုံသေမမှတ်တော့ဘဲ စကားလုံးတွေရဲ့ ဖြစ်နိုင်ခြေ ပုံသဏ္ဌာန် Pattern ကို မှတ်လိုက်တာပါ။

ဒီတော့ စကားစုတစ်စုကို နှုတ်ဆက်စကား ဟုတ်မဟုတ် သိချင်တဲ့အခါ၊ အဲ့ဒီစကားစုကို လည်း စာလုံးတွေခွဲထုတ်ပြီး ထုခြေလိုက်။ ရလာတဲ့ အထူအပါး၊ အစိတ်အကျဲက၊ ကြိုတင်မှတ်သားထားတာနဲ့ နီးစပ်ဆင်တူသလား အလွှာလိုက် တိုက်ကြည့်ပါတယ်။ တော်တော်ကြီး နီးစပ်ဆင်တူတယ်ဆိုရင် ပါဝင်တဲ့ စကားလုံးတွေ အတိအကျ တူစရာမလိုဘဲ နှုတ်ဆက်စကား ဟုတ်တယ်လို့ သိနိုင်သွားပါတယ်။

ဒါဟာ ကနေ့ခေတ် AI နည်းပညာတွေရဲ့ အလုပ်လုပ်ပုံပါပဲ။

ဒီနေရာမှာ Deep Learning တို့ Neural Network တို့လို နောက်ကွယ်က နည်းပညာတွေ အကြောင်း အသေးစိတ်ကြီး မပြောဘဲ မည်သူမဆို လိုက်ပါစဉ်းစားကြည့်လို့ ရနိုင်

လောက်တဲ့ သဘောတရားနဲ့ ပြောပြလိုက်တာပါ။ အတိအကျကြီး မဟုတ်ပေမဲ့ ဒီ ဥပမာကနေ မှတ်သားစရာတွေ ရှိပါတယ်။

- စကားလုံးတွေ ခွဲထုတ်တဲ့အလုပ်ကို Tokenization လို့ ခေါ်ပါတယ်။
- ခွဲထုတ်ရရှိလာတဲ့ စကားလုံးလေးတွေကို Token လို့ ခေါ်ပါတယ်။ Token တစ်ခုဟာ စကားလုံး တစ်လုံးလည်း ဖြစ်နိုင်တယ်။ စကားစု တစ်စုလည်း ဖြစ်နိုင်တယ်။ သင်္ကေတ တစ်ခုလည်း ဖြစ်နိုင်ပါတယ်။
- အထူအပါး၊ အနီးအဝေး၊ အစိတ်အကျဲ၊ အလွှာလိုက် ပုံသဏ္ဌာန်ခွဲ လေ့ကျင့်မှတ်သားတဲ့ အဆင့်ကို Training လို့ ခေါ်ပါတယ်။
- နောက်ဆုံးရရှိလာတဲ့ ပုံသဏ္ဌာန် နီးစပ်မှုရှိမရှိ တိုက်ဆိုင်စစ်ဆေးပေးနိုင်တဲ့ ရလဒ်ကို Model လို့ ခေါ်ပါတယ်။

အခုပေးထားတဲ့ နမူနာအရဆိုရင် Model ကို စကားစုတစ်စု ပေးလိုက်ရင် လေ့ကျင့်ထားတဲ့ စကားစုတွေနဲ့ အတိအကျ တူစရာမလိုဘဲ နှုတ်ဆက်စကား ဟုတ်မဟုတ် Model က သရုပ်ခွဲပေးနိုင်သွားမှာပဲ။

အဲ့ဒီလို သရုပ်ခွဲပေးတဲ့အခါ ဖြစ်တန်ခြေ ကို ကြည့်ပြီး ခွဲပေးတယ်ဆိုတာ အရေးကြီးတဲ့ သဘောတရား တစ်ခုပါ။ Model က ဖြစ်တန်ခြေ ဘယ်လောက်များသလဲ ဆိုတာကိုကြည့်ပြီး ခွဲပေးတာပါ။

Model က ဟုတ်မဟုတ် အတိအကျ ရာနှုန်းပြည့်ကြီး မသိပါဘူး။ ဖြစ်နိုင်ခြေရှိတယ် မရှိဘူးလို့ပဲပြောနိုင်တာပါ။ အတိအကျ ရာနှုန်းပြည့် သိနေတဲ့ Model တွေဟာလည်း အသုံးမဝင်ကြပါဘူး။ Overfitting ခေါ်ပါတယ်။ အတိအကျသိတာ ကောင်းသလိုလို နဲ့ အနည်းငယ်ကွဲပြားတဲ့ Input ကို ပေးလိုက်တာနဲ့ လုံးဝမသိနိုင်တော့ဘူး ဖြစ်သွားလို့ပါ။

ဒီနည်းနဲ့ လက်ရှိ Large Language Model (ခေါ်) LLM AI နည်းပညာတွေ မပေါ်ခင် ကတည်းက AI နည်းပညာတွေဟာ အင်တာနက်ပေါ်မှာ နေရာအနှံ့အပြား ရှိနေခဲ့တာပါ။ Online Shopping တွေက ကိုယ်ကြိုက်မယ်ထင်လို့ လာပြတဲ့ပစ္စည်းတွေ၊ YouTube တို့ ဘာတို့က ကြိုက်မယ်ထင်လို့ လာပြတဲ့ ဗွီဒီယိုတွေ၊ ဆိုရှယ်မီဒီယာပေါ်က People You May Know လို ဟာတွေဟာ ဒီ AI နည်းပညာတွေကို အသုံးပြု အလုပ်လုပ်ခဲ့ကြတာပါ။

Transformer

ဒီလိုနဲ့ (၂၀၁၇) ခုနှစ်ထဲမှာ အရာအားလုံးကို ပြောင်းလဲသွားစေတဲ့ **“Attention is All You Need”** ခေါင်းစဉ်ပါ စာတမ်းတစ်စောင်ကို Google က ပညာရှင်များက ဖော်ထုတ်ခဲ့ကြပါတယ်။

အရင်က စာသားတွေကို သရုပ်ခွဲနားလည်ဖို့ ကြိုးစားကြတဲ့ Language Model တွေဟာ စကားလုံးတွေကို တစ်လုံးချင်း ဘာပြီးရင်ဘာလာသလဲ အစီအစဉ်အတိုင်းယူ အလုပ်လုပ်ကြပါတယ်။ ဒါကြောင့် အဆက်အစပ် Context ကို သေချာနားမလည်ကြပါဘူး။

ဥပမာ - “မောင်မောင် ကျောင်းကအပြန် ကြောင်လေးနောက်လိုက်နေစဉ် ခွေးကိုက်ခံရသည်” ဆိုတဲ့စာကို ပေးလိုက်ရင် တစ်လုံးချင်း အလုပ်လုပ်သွားမှာဖြစ်လို့ “**ခွေးကိုက်ခံရသည်**” ဆိုတဲ့နေရာကို ရောက်တဲ့အခါ ခွေးက ကြောင်ကိုကိုက်တာလား၊ မောင်မောင်ကို ကိုက်တာလား Model က မသိနိုင်တော့တာ ဖြစ်နိုင်ပါတယ်။

Google ရဲ့ စာတမ်းမှာ ဖော်ပြထားတဲ့ Transformer လို့ခေါ်တဲ့ နည်းစနစ်က၊ အဲ့ဒီလို စကားလုံးတွေကို ဘာပြီးရင်ဘာလာသလဲ တစ်လုံးချင်းမယူတော့ပါဘူး။ အားလုံးကို တစ်ပြိုင်တည်း ယူပါတယ်။ ပြီးတဲ့အခါ ဘယ်စာလုံးက ဘယ်စာလုံးနဲ့ ဘယ်လောက်ထိ ပို သက်ဆိုင်သလဲ ဆိုတဲ့ အမှတ်တွေကို ပေးသွားပါတယ်။

စောစောကပြောတဲ့ မောင်မောင် ကျောင်းကအပြန် ကြောင်လေးနောက်လိုက်နေစဉ် ခွေးကိုက်ခံရသည် ဆိုတဲ့စကားစုမှာဆိုရင် Token တစ်ခုချင်းစီ အားလုံးကို အမှတ်ပေးသွားတဲ့အခါ “မောင်မောင်” နဲ့ “ခွေးကိုက်” ဟာ “ကြောင်လေး” နဲ့ “ခွေးကိုက်” ထက် ပို နီးစပ်သွားမှာ ဖြစ်ပါတယ်။ အလားတူ စကားလုံးတွေ စကားစုတွေရဲ့ အချိတ်အဆက် Pattern တွေကို အများကြီး ကြိုလေ့ကျင့် မှတ်သားထားလို့ သိတာပါ။

အဲ့ဒီလို အချိတ်အဆက်တွေကို ကြည့်ပြီး မူလစာရဲ့ ရည်ရွယ်ချက် Context ကို Model က ပိုနားလည်သွားပါတယ်။ ဒီသဘောကို Attention Mechanism လို့ ခေါ်ကြပါတယ်။

Vector တွေ Embedding တွေ Matrix Calculation တွေလို နောက်ကွယ်က တွက်ချက်ပုံ တွေ အသေးစိတ်မပြောတော့ဘဲ စိတ်ကူးနဲ့ မြင်ကြည့်လို့ရအောင် ပြောပြတာပါ။

ဒီနည်းကြောင့် Context အချိတ်အဆက် ကို Model က ပိုနားလည်သွားရုံတင် မကပါဘူး။ Training လုပ်တဲ့အချိန်မှာ စာလုံးတွေအများကြီးပေါ်မှာ တစ်ပြိုင်တည်း တွက်ချက်ပြီး Train လို့ရသွားသလို၊ ပြန်လည်အဖြေရှာတဲ့အခါမှာလည်း Input Token တွေကို တစ်ပြိုင်တည်း တွက်ချက် အဖြေရှာနိုင်သွားပါတယ်။

GPU တွေ အဓိကကျလာတာ ဒီနည်းပညာကြောင့်ပါ။

ကွန်ပျူတာတွေရဲ့ အဓိကဦးနှောက်ဖြစ်တဲ့ CPU တွေဟာ ပိုမိုရှုပ်ထွေးတဲ့ တွက်ချက်မှုတွေကို လုပ်ပေးနိုင်ကြပါတယ်။ ဒါပေမဲ့ Graphic ရုပ်ပုံ တွက်ချက်မှုတွေအတွက် သုံးတဲ့ GPU တွေက ဒီလိုမျိုး အပိုင်းလိုက်ခွဲပြီး တစ်ပြိုင်တည်း ထပ်ခါထပ်ခါ လုပ်ရတဲ့အလုပ်တွေမှာ ပိုမြန်ကြပါတယ်။ ဒါကြောင့် GPU တွေကို Graphic အတွက်တင်မဟုတ်ဘဲ AI အတွက်ပါ သုံးလာကြတာပါ။

အရင်က တွက်ချက်မှုစွမ်းအင် ဘယ်လောက်ပဲ ထပ်တိုးတိုး အစီအစဉ်အတိုင်းသွားတဲ့ Training လုပ်ပုံလုပ်နည်းမှာ အကန့်အသတ်ရှိနေလို့ သိပ်မထူးလှပါဘူး။ အခုတော့ GPU တွေ ထပ်တိုးနိုင်ရင် တိုးနိုင်သလောက် ဒေတာတွေ ပိုများများနဲ့ Train လို့ရသွားတာပါ။

တွက်ချက်မှုစွမ်းအင်တွေနဲ့ ဒေတာတွေ များသထက်များလာတဲ့အခါ ပညာရှင်များကိုယ်တိုင် အကြောင်းရင်း တိတိကျကျ မပြောနိုင်ဘဲ အံ့အားသင့်ရတဲ့ သဘောတွေလည်း ဖြစ်ပေါ်လာကြတယ်။ Emergent Behavior လို့ ခေါ်ပါတယ်။ Grokking ဆိုတဲ့ သဘောတရားလည်း ရှိပါသေးတယ်။ လိုရင်းကတော့ ဘာကြောင့်မှန်း တိတိကျကျမပြောနိုင်ဘဲ၊ ထင်မှတ်ထားတာထက် အများကြီးပိုမြင့်မားတဲ့ အပြန်အလှန် နားလည်နိုင်စွမ်း၊ ချိတ်ဆက်နိုင်စွမ်း နဲ့ အသိဉာဏ် စွမ်းဆောင်ရည်တွေ ထွက်ပေါ်လာတာပါ။

ဒီနည်းနဲ့ ကနေ့ခေတ် AI ခေတ်ပြောင်းတစ်ခုကို ရောက်ရှိသွားစေတဲ့ LLM (ခေါ်) Large Langue Model တွေ ထွက်ပေါ်လာကြတာပါ။

Generative AI

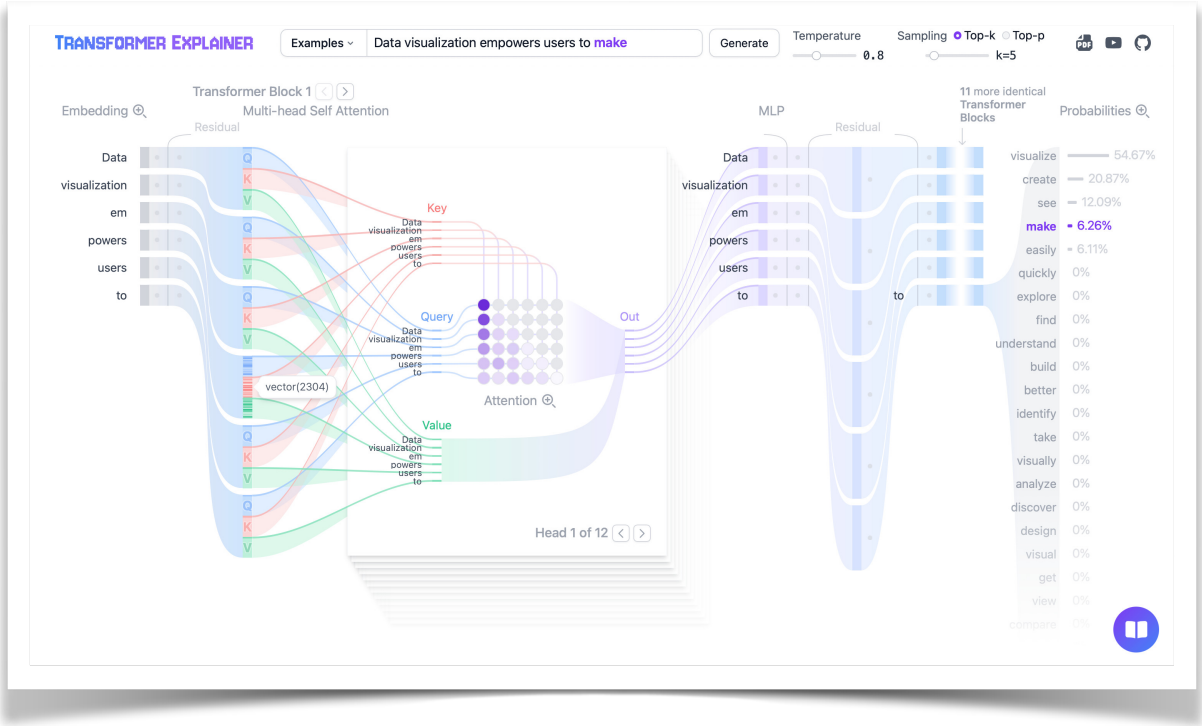
တစ်ဆက်တည်းမှာပဲ Generative AI ခေါ်ပေးလိုက်တဲ့ Input Token တွေပေါ်မှာ မူတည်ပြီး ဖြစ်နိုင်ခြေရှိတဲ့ နောက်ဆက်တွဲ Token တွေကို ရိုက်ထုတ်ပေးနိုင်တဲ့ နည်းပညာတွေပေါ်လာကြပါတယ်။

အဲ့ဒီနည်းပညာတွေထဲက တစ်ခုဟာ ChatGPT ဆိုတဲ့အမည်နဲ့ (၂၀၂၂) ခုနှစ်မှာ ထွက်ပေါ်လာပြီးနောက် အရာရာကို ပြောင်းလဲသွားစေခဲ့တာ စာဖတ်သူများလည်း သိရှိကြပြီးဖြစ်ပါတယ်။

ဒီနေရာမှာ စိတ်ဝင်စားစရာ သဘောတရားတစ်ခုကို ဖြည့်စွက်သတိပြုရပါမယ်။ ChatGPT အပါအဝင် Gemini, Claude စတဲ့ Generative AI နည်းပညာတွေဟာ Transformer Model တွေ ဖြစ်ကြလို့ ပေးလိုက်တဲ့စကားလုံးတွေရဲ့ အချိတ်ဆက်နဲ့ ရည်ရွယ်ချက် Context ကို ကောင်းကောင်းနားလည်ကြပါတယ်။ ပြီးတဲ့အခါ ဖြစ်နိုင်ခြေရှိတဲ့ နောက်ဆက်တွဲစကားလုံးတွေကို ပြန်ပေးနိုင်ကြပါတယ်။

အဲ့ဒီလိုပြန်ပေးတဲ့အခါ ဖြစ်နိုင်ခြေ အရှိဆုံး Token ကို ပြန်မပေးဘဲ ဖြစ်နိုင်ခြေရှိတဲ့ Token တွေထဲကနေ ကျပမ်းရွေးချယ်ပြီး ပြန်ပေးကြလေ့ရှိပါတယ်။ ဒီသဘောကြောင့်

ပြန်ရတဲ့ရလဒ်ဟာ မူသေကြီး ဖြစ်မနေတော့ပါဘူး။ မတူကွဲပြားတဲ့ ပေါင်းစပ်မှုတွေရရှိလာပါတယ်။ ထင်မှတ်မထားတဲ့ ဖန်တီးမှုရလဒ်တွေ ရလာပါတယ်။ နောက်ဆက်တွဲစကားလုံး တစ်လုံးပြောင်းသွားတာနဲ့ သူ့နောက်က ဆက်လာမယ့် စကားလုံးတွေပေါ်မှာလည်း ဆက်တိုက် ထပ်ဆင့် ပြောင်းလဲသွားတာ မို့လို့ပါ။



ပုံမှာမြင်တွေ့ရတဲ့ ဥပမာကို ကြည့်လိုက်ပါ။ “Data visualization empowers users to” ဆိုတဲ့ Input ကို ပေးလိုက်တဲ့အခါ နောက်ဆုံးမှာ ဖြစ်နိုင်ခြေအရှိဆုံးက **visualize** ဖြစ်ပါတယ်။ visualize ကို ရွေးလိုက်ရင် “Data visualization empowers users to **visualize**” ဖြစ်သွားမှာပါ။ လက်ရှိနမူနာမှာ **make** ကို ရွေးသွားပါတယ်။ ဒါကြောင့် “Data visualization empowers users to **make**” ဖြစ်သွားပါတယ်။ ဖြစ်နိုင်ခြေ ရှိတာတွေထဲက ရွေးသွားတယ်ဆိုတာ ဒီသဘောကို ဆိုလိုတာပါ။

ဒီပုံက Transformer အလုပ်လုပ်ပုံကို သရုပ်ဖော် ရှင်းပြထားတဲ့ နည်းပညာတစ်ခုကနေ ရတာပါ။ အောက်ကလင့်မှာ လေ့လာကြည့်လို့ရပါတယ်။

https://poloclub.github.io/transformer-explainer/

AI ကို ဘယ်လောက်ထိ ပေါင်းစပ်ဖန်တီးခွင့်ပေးမလဲ၊ Creative ဖြစ်ခွင့်ပေးမလဲဆိုတာ ကိုယ့်ဘက်က ရွေးချက်သတ်မှတ်ပေးလို့ရတဲ့ နည်းပညာတွေလည်း ရှိကြပါတယ်။ အဲ့ဒီ အကြောင်းအရာတွေကိုတော့ ဒီနေရာမှာ အသေးစိတ် အကျယ်မချဲ့နိုင်တော့ပါဘူး။

Reasoning Models

ဆက်လက်ပြီး ဖြည့်စွက်မှတ်သားသင့်တာကတော့ Chain of Thought (CoT) ခေါ် နည်းပညာကို အသုံးပြုထားတဲ့ Reasoning Model တွေပါပဲ။

ဒီ Model တွေက ရိုးရိုး Model တွေလို Input ပေါ်ကြည့်ပြီး ဖြစ်နိုင်ခြေ Output ကို တန်းပြန်မပေးပါဘူး။

- ရလဒ် အဖြေ တန်းထုတ်ဖို့ထက် Input ရဲ့ Context ဆိုလိုရင်းကို ပိုနားလည်အောင် အရင်လုပ်ဖို့ Train ထားကြပါတယ်။
- Input Context ကို ပိုနားလည်ပြီဆိုတော့မှ ဖြစ်နိုင်ခြေ ရလဒ်အမျိုးမျိုးကို မူကြမ်းထုတ်ကြည့်ပါတယ်။

- ထုတ်ကြည့်လိုက်တဲ့ မူကြမ်းရလဒ်ဟာ အချိတ်အဆက် မိမမိ၊ သဘာဝကျမကျ၊ ဖြစ်နိုင်ခြေ ရှိမရှိ၊ ပြန်ကြည့်ပါတယ်။
- သဘာဝမကျဖြစ်နေရင်၊ လွဲနေရင်၊ နောက်မူကြမ်း တမျိုး ပြောင်းထုတ်ကြည့်ပါတယ်။
- နောက်ဆုံးရလာတဲ့ ရလဒ်ကို ပိုပြည့်စုံအောင်၊ ပိုတိကျအောင် လုပ်ပါတယ်။

အဲ့ဒါတွေအားလုံးပြီးတော့မှ Output ကို ပြန်ထုတ်ပေးပါတယ်။ အဲ့ဒီမှာ Input ပေါ်မူတည်ပြီး ဖြစ်နိုင်ခြေရှိတဲ့စကားလုံးတွေ Output ထွက်လာတယ် ဆိုတာထက် အများကြီးပိုတဲ့၊ ရိုးရိုး Model တွေထက် စွမ်းဆောင်ရည်ပိုကောင်းတဲ့ Model တွေ ရလာကြပါတယ်။

ဒီ Model တွေကို Thinking Model သို့မဟုတ် Reasoning Model လို့ ခေါ်ကြပါတယ်။ အခြေခံအလုပ်လုပ်ပုံက အတူတူပါပဲ။ အဖြေကို ချက်ချင်းမပေးပဲ အထပ်ထပ် ပြန်စစ်ပြီးမှ ပေးလိုက်တာပါ။

Conclusion

ဒီအခန်းရဲ့ ရည်ရွယ်ချက်က AI နည်းပညာတွေရဲ့ အလုပ်လုပ်ပုံကို အကြမ်းဖျဉ်းနားလည်စေဖို့ ဖြစ်ပါတယ်။ ဒီလိုနားလည်ထားမှ အားသာချက်တွေ ရော အကန့်အသတ်တွေ ကိုပါသိပြီး ထိထိရောက်ရောက် အသုံးချနိုင်မှာပါ။

လက်ရှိအသုံးပြုနေကြတဲ့ Generative AI နည်းပညာတွေဟာ -

- တိတိကျကျကြိုတင်သတ်မှတ်ထားတဲ့ စည်းမျဉ်းတွေကို အခြေခံအလုပ်လုပ်တဲ့ နည်းပညာတွေ မဟုတ်ကြပါဘူး။
- Pattern ခေါ် သရုပ်သဏ္ဌာန်ကို ခွဲခြားနားနိုင်လည်နိုင်တဲ့ နည်းပညာတွေ ဖြစ်ပါတယ်။
- အတိအကျ ရာနှုန်းပြည့် အမှားအမှန် အဖြေထုတ်ပေးတဲ့ စနစ်တွေ မဟုတ်ကြဘဲ ဖြစ်တန်ခြေဘယ်လောက် များသလဲ ဆိုတဲ့ သဘောတရားပေါ်မှာ အခြေပြု အလုပ်လုပ်ကြပါတယ်။
- အတိအကျအလုပ်လုပ်ရင် အဲ့ဒီ အတိအကျအလုပ် တမျိုးတည်းကိုပဲ လုပ်တတ်ပြီး၊ အခုလို ဖြစ်တန်ခြေပေါ်မှာလုပ်လုပ်လို့ နေရာစုံမှာ အလုပ်လုပ်သွားတယ် ဆိုတာ ထူးခြားချက် တစ်ခုပါ။
- ပေးလိုက်တဲ့ စကားလုံးတွေရဲ့ အချိတ်အဆက် Context ကို ကောင်းကောင်း နားလည်ကြပါတယ်။
- ဖြစ်နိုင်ခြေရှိတဲ့ နောက်ဆက်တွဲ စကားလုံးတွေကို ပြန်ထုတ်ပေးပါတယ်။

ဒီသဘောတွေကို သိရှိထားမှ အဆင်ပြေရင်ဘာကြောင့်၊ အဆင်မပြေရင်ဘာကြောင့် ဆိုတာကို နားလည်ပြီး သင့်တော်သလို ဆက်လက် အသုံးချနိုင်မှာပါ။

အချုပ်အားဖြင့် AI ဆိုတာ ပေးလိုက်တဲ့ စကားလုံးတွေပေါ်မှာ မူတည်ပြီး ဖြစ်နိုင်ခြေကို ခန့်မှန်းပြီး ပြန်ပေးတဲ့ နည်းပညာမျှသာ ဖြစ်တယ်လို့ ဆိုနိုင်ပါတယ်။

ဥပမာ - ၁ + ၁ = ၂ ဆိုတာကို သိတဲ့ အသိဉာဏ် AI မှာရှိပါဘူး။ ၁ + ၁ ဆိုရင် နောက်ဆက်တွဲ အဖြေ (၂) ဖြစ်နိုင်ခြေ ရှိတယ်ဆိုတာကိုသာ သိတာပါ။

ဒါကြောင့် **Strawberry** ဆိုတဲ့ စာလုံးမှာ "**r**" ဘယ်နှစ်လုံးပါလဲ ဆိုတဲ့ မေးခွန်းမျိုးတွေ တိတိကျကျ မဖြေနိုင်တာပါ။ သူက စာလုံးတစ်လုံးချင်းစီ လိုက်ရေတွက်ဖို့ အသိဉာဏ်ရှိတဲ့ နည်းပညာ မဟုတ်ပါဘူး။

အလားတူပါပဲ Generative AI ကို Chess ကစားခိုင်းရင်လည်း Chess Engine တွေကို မှီမှာ မဟုတ်ပါဘူး။ Chess Engine တွေက Chess ကစားဖို့ သီးသန့် Train ထားတဲ့ AI တွေ ဖြစ်ပါတယ်။ သီးသန့်နည်းပညာဆိုရင် သက်ဆိုင်ရာကဏ္ဍမှာ အရမ်းတော်သလောက် ဘက်စုံတော့သုံးလို့မရပါဘူး။ Generative AI တွေကတော့ ပိုပြီးတော့ General ဖြစ်သွားလို့ ဘက်စုံအထွေထွေ အသုံးဝင်သွားတာပါ။ ဒီလို မတူတဲ့ကွဲပြားမှုကို မြင်ဖို့ တော်တော်လေး အရေးကြီးပါတယ်။

ဒီနေရာမှာ ဖြစ်နိုင်ခြေခန့်မှန်းပေးယုံသက်သက်ပဲ ဆိုတာကြောင့် လျှော့တွက်မိရင်တော့ မှားပါလိမ့်မယ်။ သူ့သဘောတရားပေါ်လွင်အောင်သာ ပြောလိုက်တာပါ။ လက်တွေ့မှာ Train ထားတဲ့ များပြားလှတဲ့ ဒေတာတွေ၊ Transformer လို နည်းပညာတွေ၊ အလိုလို ပေါ်လာတဲ့ Emergence လို သဘောတရားတွေကြောင့် အချိတ်အဆက် နားလည်နိုင်စွမ်းတွေ အလွန်မြင့်မားကြတာကို အားလုံးလည်း နေ့စဉ်တွေ့မြင်နေကြရမှာ ဖြစ်ပါတယ်။

AI တွေဟာ အမှန်တကယ်ပဲ၊ ကိုယ်ပိုင်အသိဉာဏ်တွေ၊ စိတ်ခံစားချက်တွေ မရှိကြပါဘူး။ ရှိတယ်လို့ ထင်ရသည့်တိုင်အောင် မရှိကြပါဘူး။ ဒါပေမဲ့လည်း အထက်မှာဖော်ပြခဲ့တဲ့ နည်းတွေကြောင့် တော်တော်ကြီးကို မြင့်မားတဲ့ ဉာဏ်ရည်ရှိနေကြပါတယ်။

ကိုယ်ပိုင်အသိဉာဏ်တွေ ခံစားချက်တွေ ရှိသယောင်ယောင် ပိုမတွက်မိဖို့လိုသလို၊ ခန့်မှန်းပြန်ထုတ်ပေးတာပါကွာ၊ ဒီထက်ဘာမှ မပိုပါဘူး ဆိုတာမျိုး လျှော့မတွက်မိဖို့လည်း လိုအပ်ပါတယ်။

အခန်း (၃) - Coding Models နှင့် Agents များ

အခုဒီစာကိုရေးသားနေစဉ် အဓိကအကျဆုံးသော ထိပ်တန်း AI Model တွေကတော့ -

- Google ရဲ့ Gemini
- OpenAI ရဲ့ ChatGPT နဲ့
- Anthropic ရဲ့ Claude

- တို့ပဲ ဖြစ်ကြပါတယ်။

xAI ရဲ့ Grok လည်း စွမ်းဆောင်ရည်ပိုင်း တန်းတူနီးပါး ကောင်းပေမဲ့ X Platform မှာပဲ အသုံးများပါတယ်။ ပြင်ပမှာ အသုံးနည်းပါတယ်။

နာမည်တွေက နည်းနည်းတော့ ရှုပ်နေပါတယ်။ Google Gemini ဆိုရင် Model အမည်က လည်း Gemini 3.0 ဖြစ်ပြီး Gemini App လည်းရှိပါတယ်။ ChatGPT ကျတော့ Model

က GPT 5.2 ဖြစ်သွားပြီး App အမည်က ChatGPT ဖြစ်နေပါတယ်။ ဒါမျိုးတွေကြောင့် နာမည်တွေ ရောပြီး တော်တော် မျက်စိရှုပ်ပါတယ်။ ကြည့်ရလွယ်အောင်၊ Gemini, ChatGPT နဲ့ Claude လို့ပဲ လိုတိုရှင်း အသုံးပြု ဖော်ပြချင်ပါတယ်။

အများအသုံးပြုကြတဲ့ AI နည်းပညာတွေထဲမှာ ChatGPT က လက်ရှိအသုံးပြုသူ သန်း (၈၀၀) ကျော်နဲ့ အသုံးအများဆုံးဖြစ်ပါတယ်။ ဒါပေမဲ့ မြန်မာနိုင်ငံမှာတော့ Gemini ကို လူသုံးအများဆုံး ဖြစ်နိုင်ပါတယ်။ Google ရဲ့ နည်းပညာဖြစ်လို့ လူတိုင်း လက်တကမ်းမှာ အလွယ်တကူ ရရှိနိုင်တာကြောင့် ပါသလို၊ မြန်မာဘာသာပိုင်းမှာ အကောင်းဆုံးဖြစ်လို့လည်း ပါပါလိမ့်မယ်။

Anthropic ရဲ့ Claude Model တွေကတော့ အစောပိုင်းမှာ စွမ်းဆောင်ရည် အမြဲတမ်း သူများထက်ရှေ့ပြေးနေလေ့ရှိတဲ့ Model တွေပါ။ Model တွေသာမက၊ Agent, MCP, Claude Code စတဲ့ အရေးပါတဲ့ ဆက်စပ်နည်းပညာတွေ အတော်များများကလည်း သူဘက်က အစပြုတာ များပါတယ်။ အထူးသဖြင့် ကုဒ်ရေးသားမှုပိုင်းမှာဆိုရင် သိသိသာသာ ပိုကောင်းတဲ့ Model တွေ ဖြစ်ခဲ့ပါတယ်။

အခုနောက်ပိုင်းမှာတော့ ဒီ Model တွေဟာ သူကပိုသာတယ်၊ ငါကပိုသာတယ် သိပ်မရှိတော့ဘဲ၊ သူအသစ်ထွက်ရင် သူကသာသွားလိုက်၊ ငါအသစ်ထွက်ရင် ငါကသာသွားလိုက် နဲ့ မတိမ်းမယိမ်း အပြိုင်အဆိုင်အနေအထားမှာ ရှိနေကြပါတယ်။

ဒီစာအုပ်က အထွေထွေ AI အသုံးပြုမှုဘက်ကို သွားမှာတော့ မဟုတ်ပါဘူး။ Vibe Coding ခေါ်ကုဒ်ရေးသားမှုပိုင်းကို အဓိကသွားမှာပါ။ ဒါကြောင့် တခြားအကြောင်းအရာတွေ သိပ်အကျယ်မချဲ့တော့ဘဲ အဓိက သဘောတရား နှစ်ခုကိုပဲ ပြောချင်ပါတယ်။

Model Variations

Gemini Model တွေမှာ Fast, Pro နဲ့ Thinking ဆိုပြီး မူကွဲတွေ ရှိကြသလိုပဲ ChatGPT မှာလည်း GPT-5.2, GPT-5.2 Thinking ဆိုပြီး မူကွဲတွေ ရှိကြပါတယ်။ Claude Model တွေမှာဆိုရင် Haiku, Sonnent, Opus ဆိုပြီး Family လိုက် မူကွဲတွေထိ ရှိပါတယ်။

ဒီလို နာမည်မူကွဲတွေက အများကြီး ရှုပ်ထွေးနေသလို၊ နောင်ကိုလည်း ပြောင်းလဲနေနိုင်ပါသေးတယ်။ ဒါကြောင့် အကျဉ်းချုပ်အားဖြင့် Model တွေမှာ အခုလို မူကွဲတွေနဲ့ လာတတ်တယ်လို့ မှတ်နိုင်ပါတယ်။

Fast - မူလအလုပ်လုပ်ပုံအတိုင်း Input ပေးလိုက်တာနဲ့ ချက်ချင်း ဖြစ်နိုင်ခြေ Output ပြန်ထုတ်ပေးတဲ့ Model တွေပါ။ အလုပ်လုပ်ပုံ အမြန်ဆုံးဖြစ်ပါတယ်။ တချို့ပေါ့ပေါ့ပါးပါး ကိစ္စတွေမှာ အသုံးပြုဖို့ သင့်တော်ပါတယ်။

Thinking - အထက်မှာပြောခဲ့တဲ့ Chain of Thoughts လုပ်ငန်းစဉ်နဲ့ Input ကို ပိုနားလည်အောင် အရင်လုပ်တယ်၊ ရလဒ်ကိုလည်း သင့်မသင့်ပြန်စစ်ပြီးမှ အဖြေထုတ်ပေးပါတယ်။ ဒါကြောင့် နည်းနည်းနှေးနိုင်ပါတယ်။ ဒါပေမဲ့ ပိုကောင်းတဲ့ ရလဒ်တွေ ပြန်ရတတ်ပါတယ်။ တချို့က Thinking လို့မခေါ်ဘဲ Pro လို့ ခေါ်ကြပါတယ်။

Deep Thinking - ရိုးရိုး Thinking လို ရလဒ်ကို သင့်မသင့် ပြန်စစ်ရုံတင်မကဘဲ၊ ရလဒ်မူကြမ်း နှစ်မျိုးသုံးမျိုးထုတ်ကြည့်၊ နှစ်မျိုးသုံးမျိုးလုံးကို ပြန်စစ်၊ အကောင်းဆုံးကို ရွေးထုတ်ပေးတဲ့ သဘောပါ။ အနှေးဆုံး ဖြစ်နိုင်ပါတယ်။ ရလဒ်အကောင်းဆုံးလည်း ဖြစ်

နိုင်ပါတယ်။ တချို့က Deep Thinking လို့မခေါ်ဘဲ High လို့ခေါ်တာ ရှိသလို Max လို့ ခေါ်တာလည်း ရှိပါတယ်။

သတိပြုရမှာကတော့ တစ်ခါတစ်ရံမှာ Thinking တွေ Deep Thinking တွေကြောင့် အတွေးခေါင် လမ်းလွဲသွားတာလည်း ဖြစ်တတ်ပါသေးတယ်။ ဒီအတိုင်း တန်းဖြေလိုက်တဲ့ Fast ရဲ့ ရလဒ်အဖြေကမှ ပိုကောင်းနေတာတွေလည်း ရှိတတ်ပါတယ်။

ပြီးတော့ Thinking Model တွေဟာ ထပ်ဆင့်အလုပ်တွေလုပ်၊ ရလဒ်ကိုပြန်ထည့်၊ ပြန်စစ်စသည်ဖြင့်လုပ်ကြလို့ Token အသုံးပြုမှု သိသိသာသာ ပိုများပါလိမ့်မယ်။ ခဏလေးနဲ့ ရထားတဲ့ အသုံးပြုခွင့် ပြည့်သွားတာမျိုး ဖြစ်နိုင်ပါသေးတယ်။

ဒါကြောင့် Thinking မှ ကောင်းတယ်၊ Deep Thinking မှ ကောင်းတယ်ဆိုတာမျိုး ပုံသေသဘောမထားဘဲ၊ လိုအပ်ချက်နဲ့ သင့်တော်နိုင်တာကို ရွေးချယ် အသုံးပြုကြရမှာ ဖြစ်ပါတယ်။

Coding Models

AI ရဲ့ Coding ပိုင်း ကျွမ်းကျင်မှုဟာ ဘာသာစကား ကျွမ်းကျင်မှုထက် အများကြီးပိုသာပါတယ်။ ဘာသာစကားဆိုတာ ပြောဆိုသူ၊ အချိန်အခါပေါ်မူတည်ပြီး စကားလုံးအတူတူကိုပဲ အဓိပ္ပာယ်အမျိုးမျိုး ဖြစ်နိုင်ပေမဲ့၊ ကုဒ်ဆိုတာ တိကျပါတယ်။ ကုဒ်တူရင် ဘယ်သူပဲရေးရေး၊ ဘယ်အချိန်ပဲဖြစ်ဖြစ် အများအားဖြင့် အဓိပ္ပာယ်တူပါတယ်။

ဘာသာစကားမှာ ဖြစ်နိုင်ခြေအတွဲအဖက်တွေ တော်တော်များပေမဲ့ ကုဒ်ဆိုတာ သတ်မှတ်ထားတဲ့ ရေးနည်းဘောင် Syntax ထဲမှာပဲ ရေးရတာပါ။ ဖြစ်နိုင်ခြေ Keyword တွေ Operator တွေဟာလည်း အကန့်အသတ်နဲ့သာ ရှိပါတယ်။

ပြီးတော့ စကားတစ်ခွန်း မှားသလား မှန်သလားဆိုတာ အခြေအနေပေါ်မူတည်ပေမဲ့ ကုဒ် မှားသလား မှန်သလားဆိုတာ အလုပ်လုပ်ရင် မှန်ပြီး၊ အလုပ်မလုပ်ရင် မှားပါတယ်။ ရှင်းပါတယ်။ ဒါကြောင့် ဘာသာစကားပိုင်းမှာ အရမ်း Smart ဖြစ်ပါတယ်ဆိုတဲ့ AI တွေဟာ Coding ပိုင်းမှာ ဒီထက်မက ပို Smart ဖြစ်ကြပါတယ်။

တချို့ Model တွေက Coding အတွက်ကို အထူးပြုဖန်တီးထားတဲ့ Model တွေပါ။ Coding Model တွေနဲ့ ရိုးရိုး Model တွေဟာ အခြေခံ Model တူကြပါတယ်။ ကွာသွားတာက Train တဲ့ အချိန်ပေးတဲ့ Parameter တွေပါ။ ဥပမာ - **for** နဲ့ **while** ဟာ နီးစပ် ဆင်တူတယ်ဆိုတာ ကုဒ်မှာပဲ ရှိပါတယ်။ ရိုးရိုး ဘာသာစကားမှာ မရှိပါဘူး။ အဲ့ဒီလို Adjustment လေးတွေလုပ်ပြီး Train သွားတာပါ။ ဒါကြောင့် သီအိုရီအရ Coding Model တွေဟာ ရိုးရိုး Model တွေထက် ကုဒ်ရေးသားမှုပိုင်းမှာ ပိုကောင်းလိမ့်မယ်လို့ ပြောလို့ရပါတယ်။

ဒါပေမဲ့ <u>လက်တွေ့ Benchmark တွေမှာ သိပ်အများကြီး မဟုတ်ဘဲ နည်းနည်းပဲ ကွာတာကိုလည်း တွေ့ရပါတယ်</u>။

ChatGPT မှာ Codex ခေါ် Coding Model သီးသန့် ရှိပါတယ်။ Claude Model တွေကတော့ Coding Model လို့ သီးသန့် နာမည်မတပ်ပေမယ့် Coding ပိုင်းမှာ အကောင်းဆုံးလို့တောင် ပြောလို့ရပါတယ်။ Gemini မှာ Coding Model သီးသန့် မရှိပေမဲ့ သူလည်း Coding ပိုင်း အဆင်ပြေတာပါပဲ။ ဒီနေရာမှာ ပြောတာတွေရောပြီး ဘာကိုဆိုလိုချင်မှန်းမ

သိ ဖြစ်သွားရင် လိုရင်းအနေနဲ့ ထိပ်တန်း Model အားလုံး Coding ပိုင်းမှာ ကောင်းကြတယ်၊ အကောင်းတကာ့ အကောင်းဆုံးက ဘာလိုဆိုတာကိုသာ ရွေးနေကြတာလို့ မှတ်နိုင်ပါတယ်။

စွမ်းဆောင်ရည်တွေက ပြောင်းလဲနေနိုင်တာဖြစ်လို့ ပုံသေတော့မမှတ်ပါနဲ့။ ဒီစာကို ရေးသားနေစဉ်မှာ အကောင်းတကာ့ အကောင်းဆုံး Coding Model နှစ်ခုကတော့ -

- GPT 5.2 Codex နဲ့
- Claude Opus 4.5 တို့ပဲ ဖြစ်ကြပါတယ်။

အတိုကောက်အားဖြင့် Codex နဲ့ Opus လို့ပဲ ဆက်လက်အသုံးပြု ဖော်ပြပါမယ်။

(**ဖြည့်စွက်ချက်** - ဒီစာအုပ်ကို ရေးသားနေဆဲမှာပဲ Codex 5.3 နဲ့ Opus 4.6 တို့ အသစ် ထပ်ထွက်သွားကြပါတယ်။ လုံးဝအပြိုင်အဆိုင်ပါပဲ။ တစ်ရက်ထဲကို မိနစ်ပိုင်းပဲခြားပြီး ကြေညာသွားကြတာပါ။)

မြန်မာပြည်မှာ Gemini ကို လူသုံးပိုများပေမဲ့ Coding ပိုင်းမှာ Gemini ကို အသုံးနည်းတယ်ဆိုတာကို သတိပြုပါ။ အသုံးနည်းလို့ စွမ်းဆောင်ရည် ညံ့တာ၊ အဆင်မပြေတာတော့ မဟုတ်ပါဘူး။ သူ့ထက်ပိုကောင်းတာတွေ ရှိနေလို့ ပိုကောင်းတာကို သုံးလိုက်ကြတဲ့သဘောပါ။

Coding Model ဆိုတာက ကုဒ်တွေရေးပေးနိုင်တယ်ဆိုရုံနဲ့ မရပါဘူး။ ကုဒ်ဖိုင်တွေ တည်ဆောက်တာ၊ Run ပေးတာလိုမျိုး Agentic Feature တွေ ကောင်းကောင်း ရမရက လည်း စကားပြောပါသေးတယ်။

Claude Model တွေရဲ့ အစောပိုင်း အားသာချက်က Model ရဲ့ စွမ်းဆောင်ရည် သက်သက်ကြောင့် မဟုတ်ဘဲ အဲ့ဒီလို Agentic Feature တွေ ကောင်းတဲ့အတွက်ကြောင့် ဖြစ်ပါတယ်။ အခုတော့ Codex ကလည်း တန်းတူ ပြေးယုံသာမက ဒီစာရေးနေချိန်မှာ နည်းနည်းလေးတောင် ပိုသာနေတဲ့ အနေအထားကို ရောက်နေတာပါ။

ဆက်လက်ပြီး Coding Agents တွေအကြောင်း ကြည့်ကြပါမယ်။

Coding Agents

Agent ဆိုတာ အချုပ်အားဖြင့် AI Model တွေကို အသုံးပြုပြီး၊ ကုဒ်တွေရေးပုံယုံသာမက လိုအပ်တဲ့ အောက်ပါ ပူးတွဲအလုပ်တွေကိုပါ လုပ်ပေးနိုင်တဲ့ နည်းပညာတွေပါ။

- လိုအပ်တာတွေ Install လုပ်တယ်၊
- ကုဒ်တွေရေးတယ်၊
- လိုအပ်တဲ့ကုဒ်ဖိုင်တွေ ဖိုဒါတွေတည်ဆောက်တယ်၊
- Linter တွေ Run ပေးတယ်၊
- Lint Error တွေရှိရင် ရှင်းပေးတယ်၊

- Build လုပ်ဖို့ လိုရင် လုပ်ပေးတယ်၊
- Build Error တွေရှိရင် ရှင်းပေးတယ်၊
- ပြီးသွားတဲ့အခါ Run ကြည့်တယ်၊
- ရလဒ် အဆင်ပြေမပြေ ကြည့်တယ်၊
- မပြေသေးရင် လိုအပ်တာတွေ ပြင်ဆင်ပြီး၊
- အထက်က အဆင့်တွေ အကုန်ပြန်လုပ်ပေးပါတယ်။

ကုဒ်ရေးယုံသာမက လုပ်စရာရှိတဲ့ Process ကြီး တစ်ခုလုံးကို အစကနေအဆုံး အကုန် လုပ်ပေးသွားတာပါ။

လိုတိုရှင်းအနေနဲ့ Linter ဆိုတာ ရေးထားတဲ့ကုဒ်မှာ အမှားရှိမရှိ အကြိုရှာပေးနိုင်တဲ့ နည်းပညာ လို့ မှတ်ပါ။ အဲ့ဒီနည်းပညာနဲ့ အမှားရှာပြီး တွေ့ရင် တစ်ခါတည်း ပြင်ပေးသွားတာပါ။

Build လုပ်တယ်ဆိုတာ ရေးထားတဲ့ကုဒ်ကို Run လို့ရတဲ့ကုဒ်ဖြစ်အောင် ပြောင်းပေးခြင်း ဖြစ်တယ်လို့မှတ်ပါ။ ရေးထားတဲ့ကုဒ်နဲ့ Run လို့ရတဲ့ကုဒ် မတူတာ ဖြစ်နိုင်ပါတယ်။ တကယ် Run တဲ့ကုဒ်ကို တိုက်ရိုက်မရေးဘဲ၊ ရေးရလွယ်ပြီး ဖတ်ရလွယ်တဲ့ ကြားခံကုဒ်ကို အရင်ရေး၊ ပြီးမှာ အဲ့ဒီ ကြားခံကုဒ်ကို Run လို့ရတဲ့ကုဒ်ဖြစ်အောင် ပြောင်းယူကြပါတယ်။ အဲ့ဒီလို ပြောင်းတဲ့အချိန်မှာ အမှားတွေ့ခဲ့ရင်လည်း တစ်ခါတည်း ပြင်ပေးတာပါ။

တချို့ Model တွေက အဲ့ဒီလို အချိန်အကြာကြီးယူ၊ အများကြီးလုပ်ရတဲ့ အလုပ်တွေကို လုပ်နိုင်ကြပြီး တချို့ Model တွေက မလုပ်နိုင်ကြပါဘူး။ အထက်မှာ ပြောခဲ့တဲ့ Codex ရဲ့ လက်ရှိ အားသာချက်က တခြား Model တွေထက် အဲ့ဒီလို အချိန်အကြာကြီးယူပြီး လုပ်ရတဲ့ အလုပ်တွေကို ပြီးဆုံးတဲ့အထိ လုပ်နိုင်တာပါ။

တချို့ Coding ကျွမ်းကျင်ရဲ့သားနဲ့ Gemini တို့ ChatGPT တို့လောက်မှာ ကုဒ်တွေ ရေးခိုင်းပြီး ဒီလို Agent နည်းပညာတွေကို အသုံးမချဖြစ်ဘူးဆိုသူတွေ အများကြီး ရှိပါတယ်။ လေ့လာထားတဲ့ ပညာတွေ နှမြောဖို့ ကောင်းပါတယ်။ လေ့လာထားတာတွေနဲ့ Coding Agent တွေကို ပေါင်းစပ်အသုံးချလိုက်ရင် အများကြီး ကောင်းသွားမှာ မို့လို့ပါ။

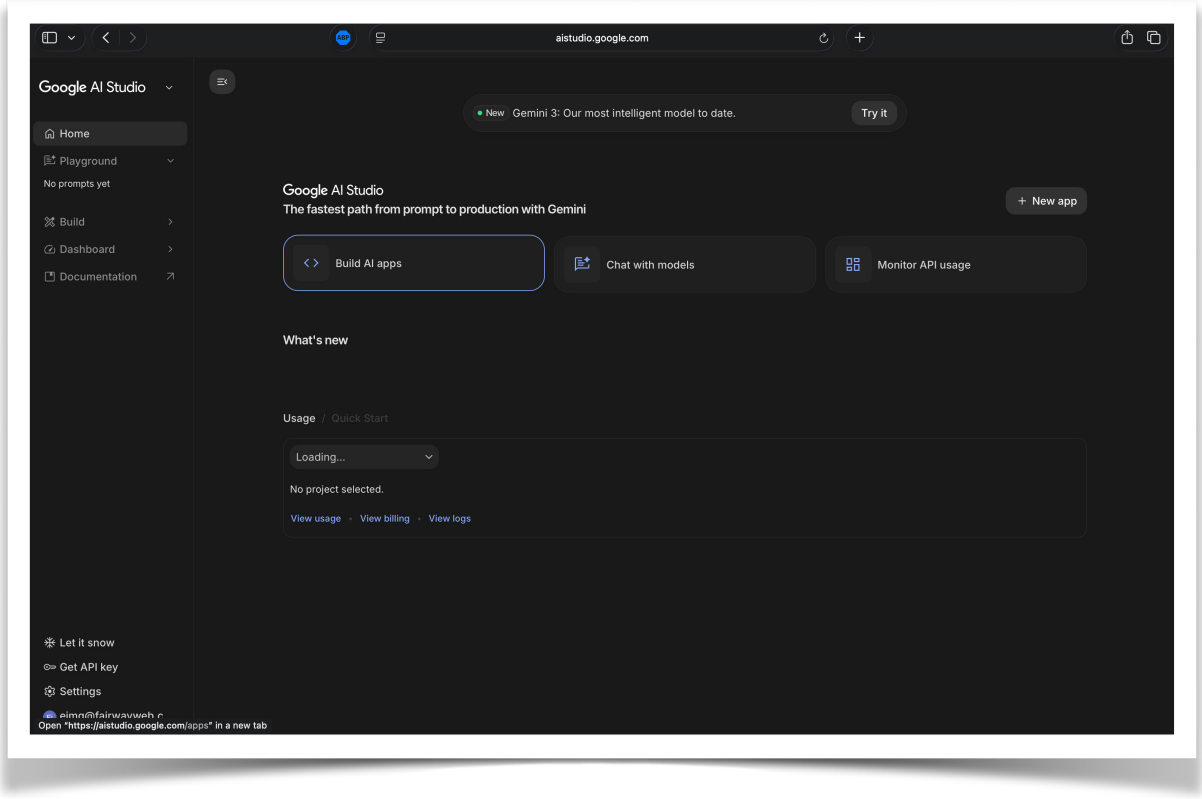
Coding Agent နည်းပညာပေါင်းများစွာ ရှိပါတယ်။ အဲ့ဒီထဲက မတူညီတဲ့ Category (၃) ခုက စာရေးသူအကြိုက် (၃) မျိုးကို ရွေးထုတ်ဖော်ပြလိုပါတယ်။

Google AI Studio

Google AI Studio ဟာ ပထမဆုံး Vibe Coding စမ်းကြည့်လိုသူတွေအတွက် အလွယ်ဆုံးနည်းလမ်းပါပဲ။ Google Account ရှိရုံနဲ့ သုံးလို့ရပြီး ကွန်ပျူတာမှာပဲဖြစ်ဖြစ်၊ ဖုန်းမှာပဲ ဖြစ်ဖြစ် အဆင်ပြေပါတယ်။

မြန်မာပြည်တွင်းက စမ်းကြည့်လိုသူတွေကတော့ VPN ဖွင့်ထားပြီး အသုံးပြုရမှာပါ။ နှစ်သက်ရာ Browser နဲ့ ဒီလင့်ကို သွားလိုက်ရုံပါပဲ။

https://aistudio.google.com



ကွန်ပျူတာမရှိသေးလို့ ဖုန်းနဲ့ပဲ စမ်းကြည့်လိုသူတွေအတွက် Replit တို့ Lovable တို့လို ဖုန်းပေါ်က Vibe Coding နည်းပညာ ရွေးချယ်စရာ အများအပြား ရှိပေမဲ့ စာရေးသူ လက်တွေ့ စမ်းကြည့်တဲ့အခါ AI Studio က တော်တော်အဆင်ပြေနေလို့ ရွေးချယ်စရာ တွေနဲ့ ခေါင်းရှုပ်ခံမနေတော့ဘဲ မြန်မာပြည်က လူတွေနဲ့ ပိုသင့်တော်နိုင်တဲ့ သူ့ကိုပဲ ပို ပြည့်စုံအောင် ထည့်သွင်းဖော်ပြတော့မယ်လို့ ရည်ရွယ်လိုက်ပါတယ်။

နောက်တစ်ခန်းမှာ AI Studio ကို အသုံးပြုပြီး Vibe Coding App လေးတချို့ အဆင့်လိုက် လုပ်ပြသွားမှာ ဖြစ်ပါတယ်။

Cursor

စာရေးသူကိုယ်တိုင် လက်တွေ့ပရောဂျက်တွေမှာ နေ့စဉ် အသုံးပြုနေတဲ့ နည်းပညာပါ။ ကျွမ်းကျင်ပရိဂရမ်မာများ ကုဒ်တွေ ကိုယ်တိုင်ရေးသားတဲ့အခါ Code Editor ဆောဖ်ဝဲတွေကို အသုံးပြုရပါတယ်။ Cursor ဆိုတာ AI Agent ပူးတွဲပါဝင်တဲ့ Code Editor နည်းပညာ ဖြစ်ပါတယ်။

ကိုယ့်ရဲ့ ကွန်ပျူတာထဲမှာ Download ရယူ Install လုပ်ပြီး သုံးရပါတယ်။ ဒီလိပ်စာမှာ ရယူနိုင်ပါတယ်။

https://cursor.com

AI ကို ကုဒ်တွေရေးခိုင်းလို့ ရယုံသာမက၊ လိုအပ်ရင် ကိုယ်တိုင်ဝင်စစ်ပြီး ဝင်ရေးလို့လည်း ရပါတယ်။ အခုနောက်ပိုင်းမှာ ကိုယ်တိုင်ဝင်စစ်ရတာ၊ ဝင်ရေးရတာတွေ နည်းသထက် နည်းလာလေပါပဲ။ ဒါပေမဲ့ ရံဖန်ရံခါ AI က ဘယ်လိုမှ မှန်အောင် လုပ်ပေးနိုင်တော့လို့ ကိုယ်တိုင် မဖြစ်မနေ ဝင်လုပ်ပေးရတာတွေက ရှိနေဆဲပါပဲ။ အဲ့ဒီလို အခြေအနေတွေမှာ အထူးအသုံးဝင်ပါတယ်။

Codex, Opus စသည်ဖြင့် ထိပ်တန်း Model တွေ သူထဲမှာ အကုန်ပါလို့ နှစ်သက်ရာကို အဆင်သင့် ရွေးချယ်အသုံးပြုလို့ရတာလည်း အားသာချက်တစ်ခုပါပဲ။ Hobby Plan နဲ့လည်း Free အသုံးပြုလို့ရပြီး တစ်လကို ဒေါ်လာ (၂၀) ပေးရတဲ့ Plan လောက် ဝယ်ထားလိုက်ရင် တော်တော်သုံးလို့ ရပါတယ်။ **အခန်း (၆)** မှာ Cursor ကို အသုံးပြုပြီး နမူနာထည့်လုပ်ပြသွားမှာပါ။

သူနဲ့အမျိုးအစားတူ ရွေးချယ်စရာတွေ အများကြီးရှိပါတယ်။

- VS Code + Github Copilot
- Google Antigravity
- Windsurf

- စသည်ဖြင့်။

AI နည်းပညာတွေဟာ Token အကန့်အသတ်နဲ့သာ အသုံးပြုခွင့်ပေးနိုင်ကြတာကို စာဖတ်သူများ သိရှိကြပြီးဖြစ်မှာပါ။ တစ်ရက်တာ၊ တစ်လတာ၊ အသုံးပြုမှုပမာဏ ကျော်သွားရင် သတ်မှတ်အချိန်ပြည့်အောင် စောင့်ပြီးမှပဲ ပြန်သုံးလို့ရပါတယ်။

ဒါကြောင့် စာရေးသူကတော့ ကွန်ပျူတာထဲမှာ အထက်မှာပြောထားတဲ့ Code Editor Agent တွေ အကုန် Install လုပ်ထားပြီး အလုပ်များလို့ Token Limit မလောက်တဲ့အချိန်တွေဆိုရင် တစ်ခုပြီးတစ်ခုပြောင်း လှည့်ပတ်သုံးပါတယ်။ ဒီစာအုပ်ရဲ့ နောက်ဆုံးခန်းမှာလည်း တခြားအခမဲ့ (သို့) မခမဲ့နီးပါး ရနိုင်တဲ့ နည်းပညာတွေ ဖြည့်စွက် ဖော်ပြထားပါသေးတယ်။

Codex

Codex ဆိုတာ OpenAI ရဲ့ Model အမည်ဖြစ်သလို အဲ့ဒီအမည်နဲ့ပဲ Coding Agent လည်း ရှိနေတာပါ။ သူက CLI Agent ခေါ် Command Line Interface နဲ့ Terminal ထဲမှာ အသုံးပြုရပါတယ်။ ဒါကြောင့် Command တွေ Terminal တွေ သေချာသုံးတတ်တဲ့ နည်းပညာပိုင်း ကျွမ်းကျင်တဲ့သူတွေအတွက် ပိုအဆင်ပြေပါတယ်။

https://chatgpt.com/codex

Codex ဟာလည်း စာရေးသူ နေ့စဉ်အသုံးပြုနေတဲ့ နည်းပညာတစ်ခုပါပဲ။ မူလနည်းပညာက Terminal ထဲမှာ သုံးရတဲ့ CLI နည်းပညာဆိုပေမဲ့ Code Editor တွေနဲ့လည်း တွဲသုံးလို့ရပါတယ်။ ဒါကြောင့် စာရေးသူက အခုလိုလည်း တွဲသုံးပါတယ် -

- VS Code + Codex

အခုဒီစာကို ရေးနေတဲ့ နေ့မှာပဲ **Codex App** ဆိုပြီး macOS အတွက် App ထုတ်ပေးလာပါတယ်။ စမ်းကြည့်ဖြစ်ပါတယ်။ အဆင်ပြေပေမယ့်၊ လက်ရှိ VS Code Editor နဲ့ Codex ကို တွဲသုံးရတာလည်း အဆင်ပြေနေလို့ အဲ့ဒီအတိုင်းပဲ ဆက်သုံးဖြစ်ပါတယ်။

Codex ထက် လူကြိုက်ပိုများတဲ့ အလားတူ နည်းပညာက **Claude Code** လို့ ခေါ်ပါတယ်။ ဒါပေမဲ့ မြန်မာပြည်ကနေ အသုံးပြုခွင့်မပေးပါဘူး။ သူတို့ဘက်က ပိတ်ထားပါတယ်။ သုံးချင်ရင် VPN ကျော်လိုက်ရုံနဲ့ မပြီးဘဲ ပြည်ပလိပ်စာတွေ ဘာတွေနဲ့ သုံးရလို့၊ တခြားအဆင်ပြေတာတွေ ရှိနေချိန်မှာ အဲ့ဒီလောက်ကြီး အလုပ်ရှုပ်ခံ မသုံးဖြစ်ပါဘူး။

https://claude.com/product/claude-code

Claude Code တို့ Codex တို့လို နည်းပညာတွေက အများကြီး ပိုကြမ်းသွားတယ်လို့ ဆိုနိုင်ပါတယ်။ အရာရာကို AI Agent က အဓိက ကိုင်တွယ်သွားတာဖြစ်ပြီး လိုအပ်ရင် ကိုယ့်ကွန်ပျူတာစနစ်ထိ ကိုယ့်ခွင့်ပြုချက်နဲ့ ဝင်ရောက်ကိုင်တွယ်နိုင်သွားတာပါ။ ဒါကြောင့် ကျွမ်းကျင်သူတွေက တော်တော်ကြိုက်ကြပါတယ်။

Gemini CLI ခေါ် Google က ပေးထားတဲ့ အလားတူ နည်းပညာလည်း ရှိသေးပေမဲ့ လူသုံးနည်းတာကို တွေ့ရပါတယ်။

https://geminicli.com/

စာရေးသူ ရံဖန်ရံခါ သုံးဖြစ်ပေမဲ့ အမြဲမသုံးဖြစ်ပါတယ်။ အဆင်မပြေလို့ မဟုတ်ပါဘူး။ ဟိုးအထက်မှာ ပြောခဲ့သလိုပဲ၊ ပိုကောင်းတာတွေ ရှိနေတော့ ပိုကောင်းတာတွေကိုပဲ သုံးဖြစ်လိုက်တဲ့ သဘောပါ။

နောက်ထပ် အလားတူ နည်းပညာထဲမှာ စိတ်ဝင်စားဖို့ကောင်းပြီး လူကြိုက်အရမ်းများနေတဲ့ OpenCode လို့ခေါ်တဲ့ နည်းပညာလည်း ရှိပါသေးတယ်။ သူ့အကြောင်းကို နောက်ဆုံးခန်းရောက်တော့မှ ထည့်သွင်းဖော်ပြပါတော့မယ်။

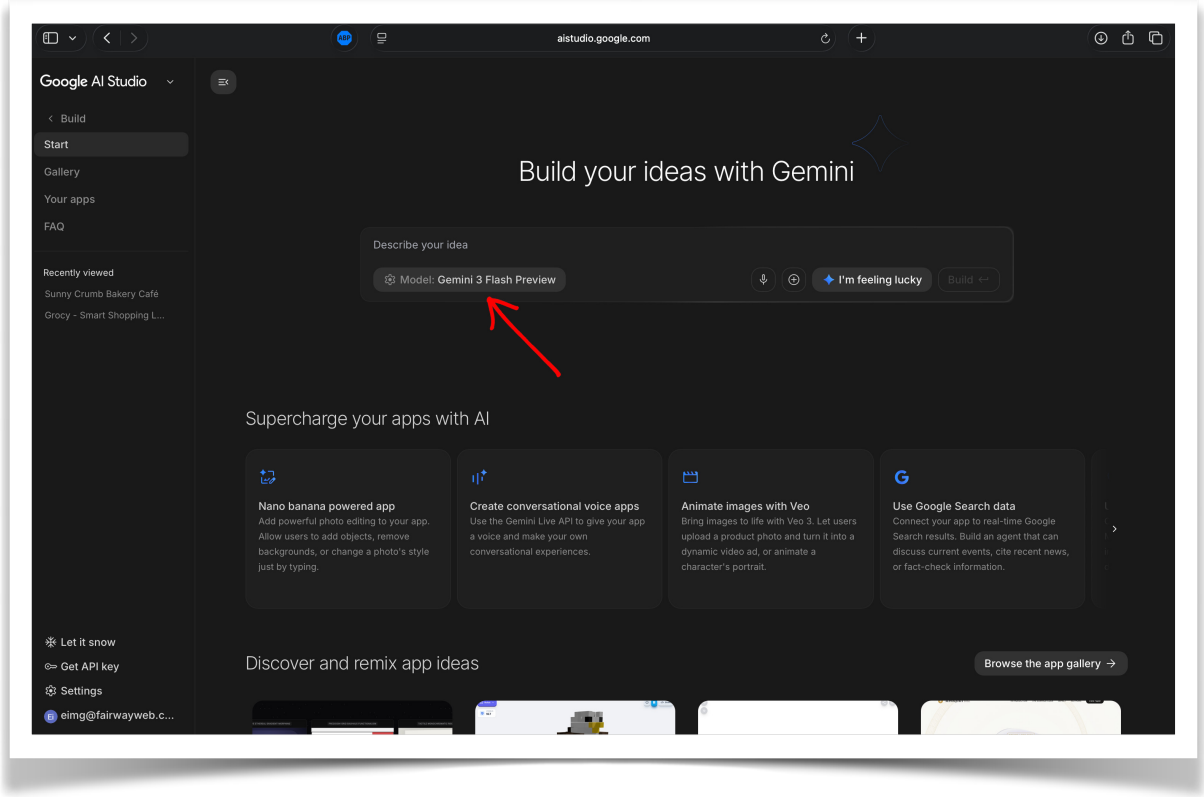
အခန်း (၄) - Vibe Coding နှင့် App များဖန်တီးခြင်း

သိသင့်တာလေးတွေ ပြောပြီးပြီဖြစ်လို့ ဒီအခန်းမှာ နမူနာ App လေးတချို့ စလုပ်ကြည့်ချင်ပါတယ်။ ဒီအခန်းမှာတော့ လူတိုင်းလိုက်လုပ်လို့ရအောင် Google AI Studio ကိုပဲ အသုံးပြုကြမှာပါ။ ကိုယ့်ကွန်ပျူတာ (သို့မဟုတ်) ဖုန်းမှာ VPN လေး ဖွင့်ထားပြီး အောက်ပါလိပ်စာကို သွားလိုက်ပါ။

https://aistudio.google.com

လိုအပ်ရင် Gmail အကောင့်နဲ့ Login ဝင်ပေးလိုက်ပါ။ AI Studio ပွင့်မလာဘဲ “**Available Regions for Google AI Studio**” ဆိုတာမျိုး အသိပေးချက် လာပြနေရင် VPN လိုအပ်နေတာပါ။

AI Studio ပွင့်လာရင်တော့ **Build AI Apps** ကို နှိပ်လိုက်ပါ။ အခုလိုပုံစံ ရပါလိမ့်မယ်။



AI App တွေမှာ တွေ့နေကြအတိုင်း မေးချင်တာတွေ့ မေးလို့ရပြီး ခိုင်းချင်တာတွေ့ ခိုင်းလို့ရတဲ့ ရိုက်ထည့်စရာ Prompt Box တစ်ခုရှိပါတယ်။

မျှားပြထားတဲ့နေရာကို နှိပ်ပြီး အသုံးပြုလိုတဲ့ Model ရွေးလို့ရပါတယ်။ စာရေးသူကတော့ **Gemini 3 Flash** ကိုပဲ ရွေးထားပါတယ်။ အသုံးပြုခွင့်အကန့်အသတ် ခဏလေးနဲ့ ပြည့်သွားတာမျိုး မဖြစ်စေချင်လို့ Pro ကို မရွေးထားပါဘူး။

အကယ်၍ စာဖတ်သူက စမ်းကြည့်ချင်တယ်ဆိုရင်လည်း Pro တွေဘာတွေ ရွေးပြီး စမ်းကြည့်လို့ရပါတယ်။ လက်ရှိ Gemini 3 က နောက်ဆုံးဖြစ်ပြီး သူ့ရှေ့က 2.5 လည်း မဆိုးပါဘူး။ 3 ကပိုကောင်းပေမဲ့ မဖြစ်မနေ လိုအပ်ရင် 2.5 လည်း သုံးလို့ရပါတယ်။

Prompts

ပထမဆုံးအနေနဲ့ ရိုးရှင်းတာလေးတစ်ခုလောက်ကနေ စလုပ်ကြည့်ချင်ပါတယ်။ အသစ်ဖွင့်မဲ့ ကိုယ့်ရဲ့ ကော်ဖီဆိုင်လေးအတွက် ချစ်စဖွယ် Website လေး တစ်ခုလောက် လိုချင်တယ် ဆိုကြပါစို့။ အဲ့ဒါလေး လုပ်ခိုင်းကြည့်ပါမယ်။

စာမျက်နှာတစ်ခုပဲပါတဲ့ ရိုးရိုးရှင်းရှင်းလေးပဲ လုပ်မှာမို့လို့ ဒီနေရာမှာ System Thinking တွေဘာတွေ မလိုအပ်သေးပါဘူး။ ဒါပေမဲ့ လိုရင်းကဘာလဲ ဆိုတာတော့ ရှင်းအောင်ပြောရမှာ ဖြစ်ပါတယ်။

- ချစ်စဖွယ်အသွင်အပြင်လေးဖြစ်ရမယ်၊
- တစ်မျက်နှာထဲမှာ အကုန်ပြပေးရမယ်၊
- ကော်ဖီနဲ့မုန့်စျေးနှုန်းလေးတွေ ပြပေးရမယ်၊
- Set Menu လေးတချို့ ပြပေးရမယ်။

ဒီလောက်ဆိုရင် ရပါပြီ။ ဒါကြောင့် စာရေးသူက အခုလို Prompt လေး ပေးလိုက်ပါတယ်။

> *I want to create a lovely landing page for my new bakery cafe. Only one page where customer can quickly see available items. Only small set of coffee, bread and menu.*

Prompt ကောင်းရင်ကောင်းသလို ရလဒ်ကောင်းပါတယ်။ Prompt ပေးတဲ့နေရာမှာ အရေးအကြီးဆုံးအချက်က တစ်ချက်တည်းပဲ ရှိတာပါ။ ရိုးရိုးရှင်းရှင်းနဲ့ ဆိုလိုရင်းကို ပေါ်လွင်ရမှာပါ။ စာလုံးပေါင်းတွေ Grammar တွေက ပြဿနာမဟုတ်ပါဘူး။ ကိုယ်ဘာကို ဆိုလိုတာလဲဆိုတာ ပေါ်လွင်ရှင်းလင်းဖို့ကသာ အဓိက ဖြစ်ပါတယ်။

တခြား Instruction တို့ Role Playing တို့လို Prompt Engineering နည်းစနစ်တွေကို ဒီနေရာမှာ မေ့ထားလိုက်လို့ရပါတယ်။ လက်တွေ့မှာ ဒီလောက်ကြီး အသုံးမဝင်ကြပါဘူး။

ရလဒ်မကောင်းရင် ဖြစ်နိုင်ခြေ နှစ်ခုပဲ ရှိပါတယ်။

- ပထမတစ်ခုက၊ ပါသင့်တဲ့ Context ပြည့်စုံအောင်မပါလို့ AI က ခန့်မှန်းဖြည့်စွက်အလုပ်လုပ်သွားတဲ့အခါ အများအားဖြင့် ကိုယ်ဖြစ်စေချင်တာနဲ့ ကိုက်ညီသွားပေမဲ့ တစ်ခါတစ်လေ မကိုက်ညီဘူး ဖြစ်သွားတာပါ။

- နောက်တစ်ခုကတော့ ကိုယ့်ဆိုလိုရင်းနဲ့ AI နားလည်လိုက်တာ မတူဘဲ Misalignment ဖြစ်သွားတာပါ။ အများအားဖြင့် AI က ဆိုလိုရင်းကို နားလည်လေ့ရှိပေမဲ့ တစ်ခါတစ်ရံ သုံးလိုက်တဲ့ စကားလုံးအသုံးအနှုန်းကြောင့်ပဲ ဖြစ်ဖြစ်၊ ကိုယ်ဆိုလိုတာကတမျိုး၊ သူနားလည်တာကတမျိုး ဖြစ်သွားတတ်ပါတယ်။

အဲ့ဒီနှစ်ခု မဖြစ်အောင်ပဲ ဂရုစိုက်ရမှာ။

ဒါတွေကြောင့် ရှေ့အခန်းတွေမှာ AI ရဲ့ အလုပ်လုပ်ပုံ ကို အရင်ကြိုရှင်းပြခဲ့တာ ဖြစ်ပါတယ်။ AI ရဲ့ အလုပ်လုပ်ပုံကို နားလည်ထားသူဟာ၊ ဖြစ်နိုင်တာကို သိတယ်၊ မဖြစ်နိုင်တာကို သိတယ်၊ ဖြစ်တတ်တာကိုသိတယ်။ ဒါကြောင့် ပိုမိုကောင်းမွန်တဲ့ Prompt တွေကို ပေးနိုင်သွားပါတယ်။

စာရေးသူက အင်္ဂလိပ်ဘာသာနဲ့ပဲ Prompt ပေးထားပါတယ်။ Gemini AI က မြန်မာစာတော်တော်ရပေမဲ့ အင်္ဂလိပ်စာ ကျွမ်းသလောက်တော့ မြန်မာစာကို မကျွမ်းပါဘူး။ ဒါကြောင့် ရလဒ်ပိုကောင်းချင်တဲ့အတွက် အင်္ဂလိပ်ဘာသာနဲ့ပဲ ပေးထားပါတယ်။

ဒါဟာ မြန်မာဘာသာနဲ့ မပေးသင့်ဘူးလို့ ဆိုလိုတာတော့ မဟုတ်ပါဘူး။ စာဖတ်သူက မြန်မာလိုပဲ Prompt ပေးချင်တယ်ဆိုရင်လည်း ပေးလို့ရပါတယ်။ ထုံးစံအတိုင်း ဖြစ်နိုင်တဲ့ သဘောတရားကို ကြိုမြင်ထားဖို့ပဲ လိုတာပါ။

တခြား Prompt Engineer နည်းစနစ်တွေ သိပ်အသုံးမဝင်ပေမဲ့ အသုံးဝင်တာလေး တစ်ခုတော့ ရှိပါတယ်။ အဲ့ဒါက AI ကို ကိုယ့်အတွက် Prompt ရေးပေးခိုင်းခြင်း ဖြစ်ပါတယ်။ ဒါကြောင့် စာရေးသူက အထက်က Prompt ကို AI Studio မှာ တိုက်ရိုက်မပေးတော့ဘဲ ChatGPT ကိုအရင်ပေးပြီး Prompt အပြည့်အစုံ အခုလို ရေးပေးခိုင်းလိုက်ပါတယ်။

> *I want to create a lovely landing page for my new bakery cafe. Only one page where customer can quickly see available items. Only small set of coffee, bread and menu.* ***Help me create a prompt to give to coding agent.***

အဲ့ဒီမှာ ChatGPT က အခုလို စာမျက်နှာ (၃) မျက်နှာစာလောက်ရှိတဲ့ လုံးဝပြီးပြည့်စုံတဲ့ Detail Prompt ကို ပြန်ပေးပါတယ်။

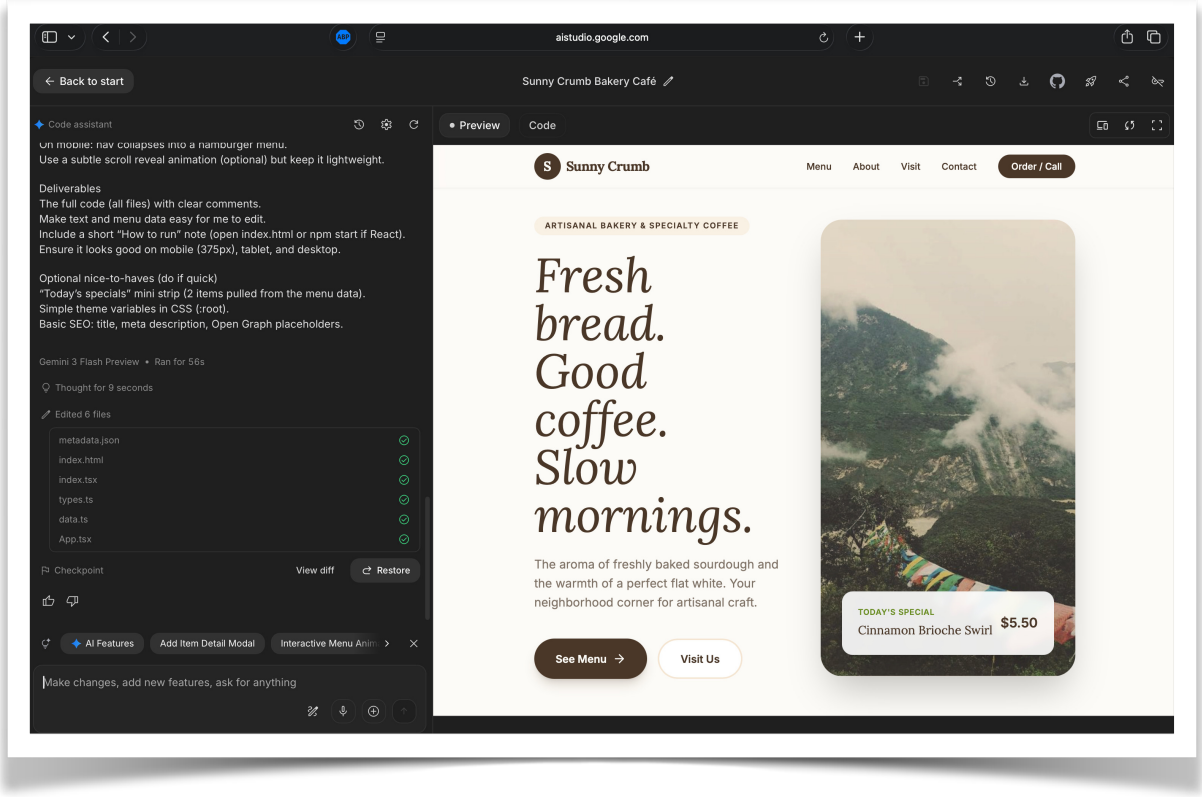
> ***One-page bakery café landing page***
>
> *Build a single-page, responsive landing page for a new bakery café. The goal is: customers quickly see what we sell (small menu), get the vibe, and know how to visit/order.*
>
> ***Brand / vibe***
>
> ***Mood**: warm, cozy, artisan, “fresh-baked morning”.*
>
> ***Palette**: cream / wheat / coffee brown + a subtle accent (muted green or soft rose). Rounded cards, soft shadows, lots of whitespace. Friendly typography (clean readable body, slightly stylized heading).*
>
> *…*

အပြည့်အစုံကိုတော့ ထည့်မပြထားပါဘူး။ ကိုယ်တိုင်အလားတူ စမ်းကြည့်လိုက်ပါ။ သုံးရမယ့် Style တွေ၊ အရောင်တွေ၊ ပါရမယ့် ကော်ဖီ နဲ့ မုန့် Item တွေ၊ ဈေးနှုန်းတွေကအစ အသေးစိတ်ပါသွားလို့ အများကြီး ပိုပြည့်စုံသွားပါတယ်။

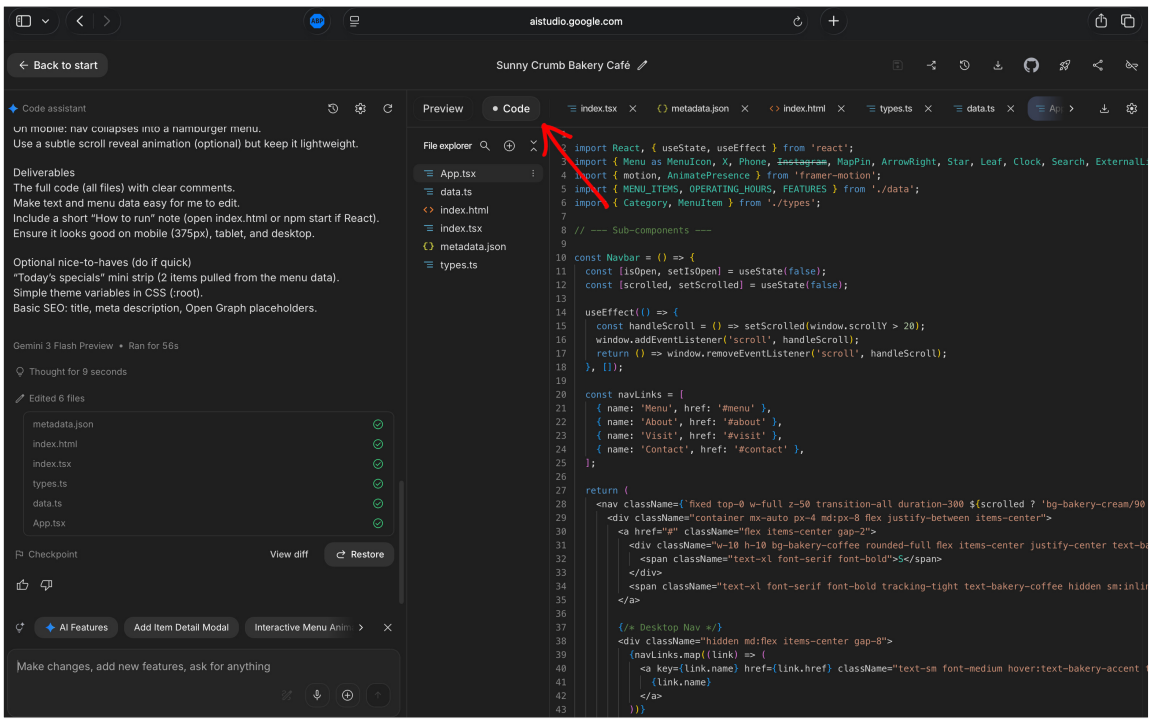
ဒီအတိုင်း အမြဲလုပ်သင့်တယ်လို့ မဆိုလိုပါဘူး။ မူလ လိုတိုရှင်း Prompt ကိုပဲ တိုက်ရိုက်ပေးလိုက်ရင်လည်း ရသလို။ အခုလို AI အကူအညီနဲ့ပဲ Detail Prompt အရင် ယူလို့လည်း ရတာပါ။ နှစ်မျိုးလုံး အသုံးဝင်ပါတယ်။

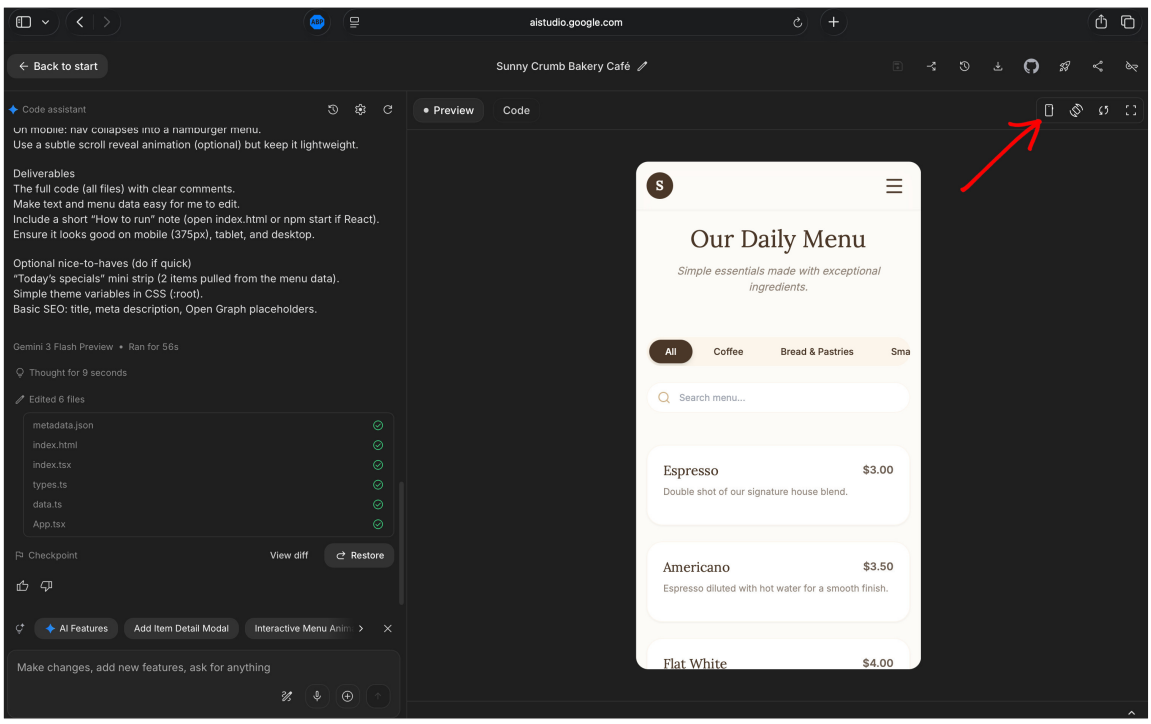
အထူးသဖြင့် ပထမဆုံး Prompt တွေမှာ Detail Prompt ဖြစ်သင့်ပြီး နောက်ဆက်တွဲ Prompt တွေမှာတော့ လိုတိုရှင်းပဲဆက်ပေးလိုက်ရင် ရပါတယ်။

စာရေးသူဆီမှာတော့ အခုလို တစ်ချက်တည်းနဲ့ လုံးဝစိတ်တိုင်းကျတဲ့ ကော်ဖီဆိုင် Website လေးတစ်ခုကို ရလိုက်ပါတယ်။ ဘယ်ဘက်ခြမ်းက AI ရဲ့အကြောင်းပြန်မှု ဖြစ်ပြီး ညာဘက်ခြမ်းက ရလဒ်ဖြစ်ပါတယ်။ တော်တော့်ကို ကောင်းမွန်သပ်ရပ်တဲ့ ဒီဇိုင်းနဲ့ ပါသင့်တာတွေ တစ်မျက်နှာထဲမှာ ပြည့်ပြည့်စုံစုံပါဝင်ပါတယ်။



ဒီရလဒ်ရဖို့အတွက် သူရေးပေးသွားတဲ့ကုဒ်တွေကိုလည်း ဖွင့်ကြည့်လို့ရပါတယ်။


aistudio.google.com
Back to start
Sunny Crumb Bakery Café
Code assistant
Use a subtle scroll reveal animation (optional) but keep it lightweight.
Deliverables
The full code (all files) with clear comments.
Make text and menu data easy for me to edit.
Include a short "How to run" note (open index.html or npm start if React).
Ensure it looks good on mobile (375px), tablet, and desktop.
Optional nice-to-haves (do if quick)
"Today's specials" mini strip (2 items pulled from the menu data).
Simple theme variables in CSS (:root).
Basic SEO: title, meta description, Open Graph placeholders.
Gemini 3 Flash Preview • Ran for 56s
Thought for 9 seconds
Edited 6 files
metadata.json
index.html
index.tsx
types.ts
data.ts
App.tsx
Checkpoint
View diff
Restore
AI Features
Add Item Detail Modal
Make changes, add new features, ask for anything
Preview
Code
File explorer
App.tsx
data.ts
index.html
index.tsx
metadata.json
types.ts


aistudio.google.com
Back to start
Sunny Crumb Bakery Café
Code assistant
Use a subtle scroll reveal animation (optional) but keep it lightweight.
Deliverables
The full code (all files) with clear comments.
Make text and menu data easy for me to edit.
Include a short "How to run" note (open index.html or npm start if React).
Ensure it looks good on mobile (375px), tablet, and desktop.
Optional nice-to-haves (do if quick)
"Today's specials" mini strip (2 items pulled from the menu data).
Simple theme variables in CSS (:root).
Basic SEO: title, meta description, Open Graph placeholders.
Gemini 3 Flash Preview • Ran for 56s
Thought for 9 seconds
Edited 6 files
metadata.json
index.html
index.tsx
types.ts
data.ts
App.tsx
Checkpoint
View diff
Restore
AI Features
Add Item Detail Modal
Make changes, add new features, ask for anything
Preview
Code
Our Daily Menu
Simple essentials made with exceptional ingredients.
All
Coffee
Bread & Pastries
Search menu...
Espresso
$3.00
Double shot of our signature house blend.
Americano
$3.50
Espresso diluted with hot water for a smooth finish.
Flat White
$4.00

အပေါ်ညာဘက်နားမှာ Share တို့ Download တို့လို ခလုတ်တွေလည်း ပါဝင်ပါတယ်။ နမူနာမှာ မျှားပြထားတဲ့ခလုတ်ကို နှိပ်ပြီး ဖုန်းမှာတွေ့ရမယ့် View တွေ Tablet မှာ တွေ့ရမယ့် View တွေနဲ့လည်း ကြည့်လို့ရပါတယ်။ ရိုးရိုးလေးနဲ့ တစ်ချက်ထဲနဲ့ လက်တွေ့အသုံးဝင်တဲ့ ရလဒ်တစ်ခုကို ရရှိလိုက်တာပါ။ ဘာမှတောင် ထပ်ညှိစရာ မလိုလိုက်ပါဘူး။

Grocery App

နောက်တစ်ဆင့်အနေနဲ့ နည်းနည်းလေး ပိုကျယ်ပြန့်တဲ့ နမူနာလေး ဆက်လုပ်ကြည့်ကြပါမယ်။ ဒီတစ်ခါ ဈေးဝယ်သွားတဲ့အခါ ပစ္စည်းတွေ မေ့မကျန်ခဲ့အောင် မှတ်တဲ့ App လေး လုပ်ကြမှာပါ။

ထုံးစံအတိုင်း ပထမဆုံး ရှင်းရမှာက၊ လိုရင်းကဘာလဲ ဆိုတဲ့အချက် ဖြစ်ပါတယ်။

- ဝယ်ရမယ့်ပစ္စည်းတွေ ထည့်မှတ်လို့ရချင်တယ်၊
- မှတ်ချက် Note လေးနဲ့ တွဲမှတ်လို့ရချင်တယ်။

System Thinking စလိုလာပါပြီ။ Input, Process, Storage, Network, Output ဘာတွေ လိုနိုင်လဲ။

- Input အနေနဲ့ ဝယ်ရမယ့်ပစ္စည်း နဲ့ Note ရိုက်ထည့်လို့ရရမယ်၊
- ခလုတ်နှိပ်လိုက်ရင် ကိုယ့်ဖုန်းထဲမှာပဲ သိမ်းသွားရမယ်၊
- မှားထည့်မိရင်၊ မလိုအပ်တော့ရင် ပြန်ဖျက်လို့ရချင်တယ်၊

- ဝယ်ပြီးသွားရင် ပြန်မဖျက်သေးဘဲ ပြန်ကြည့်လို့ရအောင် Check လုပ် မှတ်ထားလို့ ရချင်တယ်၊
- ပြန်ပြတဲ့အခါ မှတ်ထားတဲ့စာရင်းရော ဝယ်ပြီးသားစာရင်းရော အကုန်ပြန်ပြရမယ်။

ဒါတွေအကုန် AI ကို မပြောလဲရပါတယ်။ ဒါပေမဲ့ ကိုယ့်ဘက်ကတော့ အခုလို ကိုယ်ဘာလိုချင်တာလဲ ကိုယ့်ဘာသာ အရင်ရှင်းဖို့လိုပါတယ်။ ကိုယ်ဘာသာ အရင်မရှင်းဘဲ၊ ဒီအတိုင်းလုပ်ကြည့်လိုက်၊ ရလာမှ အဆင်ပြေမပြေ ကြည့်မယ်ဆိုရင် အဲ့ဒါ အဆင်မပြေမှုတွေရဲ့ အစပါပဲ။

ဒီအချက်တွေပေါ်မှာ အခြေခံပြီး အခုလိုပဲ လိုတိုရှင်း Prompt လေးပေးလိုက်ရင် ရပါတယ်။

> *When I go shopping, I always forget something. I want to create a grocery items app that I can add items when something is needed. So when the day of shopping I will not forget anything. Each item should have optional note.*

ဒါမှမဟုတ် အထက်က နမူနာမှာလို ဒီ Prompt ကိုပေးပြီး Detail Prompt AI ကို အခုလို ရေးခိုင်းလိုက်လို့ရပါတယ်။

> *When I go shopping, I always forget something. I want to create a grocery items app that I can add items when something is needed. So when the day of shopping I will not forget anything. Each item should have optional note.* ***Help me create a prompt to give to coding agent. Do not add your remark, just ready to copy prompt.***

စာရေးသူဆီမှာ အခုလို စာမျက်နှာ (၅) မျက်နှာစာလောက်ထိရှိတဲ့ Input, Process, Output တွေ အကုန် အတိအကျ သတ်မှတ်ပေးထားတဲ့ Detail Prompt ကြီး ပြန်ရလာပါတယ်။

> *Build a cross-platform grocery items app (mobile-first; web optional) that solves: “I add items anytime during the week, then on shopping day I don’t forget anything.” Implement end-to-end: codebase, UI, storage, and basic tests.*
>
> ***Core requirements***
>
> *1)* ***Items***
>
> *Create grocery items with:*
>
> ***name*** *(required, string)*
>
> ***note*** *(optional, string; can be empty)*
>
> *…*
>
> *2)* ***Shopping flow***
>
> *Main list is a checklist:*
>
> ***Tap to check/uncheck***
>
> ***Checked items move to bottom***
>
> *…*

အကုန်ထည့်ပြမထားပါဘူး။ စာဖတ်သူကိုယ်တိုင်ပဲ အထက်က လိုတိုရှင်း Prompt ကို သုံးပြီး Detail Prompt တောင်းယူကြည့်လိုက်ပါ။

Context Window

ဒီနေရာမှာ ကြားဖြတ်ပြောဖို့ လိုလာတာက Context Window အကန့်အသတ်ပါပဲ။ AI နည်းပညာတိုင်းမှာ လက်ခံနိုင်တဲ့ အမြင့်ဆုံး Token ပမာဏ ရှိပါတယ်။ Gemini က Token (၁) သန်းထိ လိုက်ခံနိုင်ပါတယ်။ ဆိုလိုတာက လက်ရှိ လုပ်နေတဲ့ ပရောဂျက်၊ Session မှာ ကိုယ်ပေးလိုက်တဲ့ Input Token နဲ့ သူပြန်ထုတ်ပေးတဲ့ Output Token စုစုပေါင်း (၁) သန်းထိ လက်ခံတာပါ။

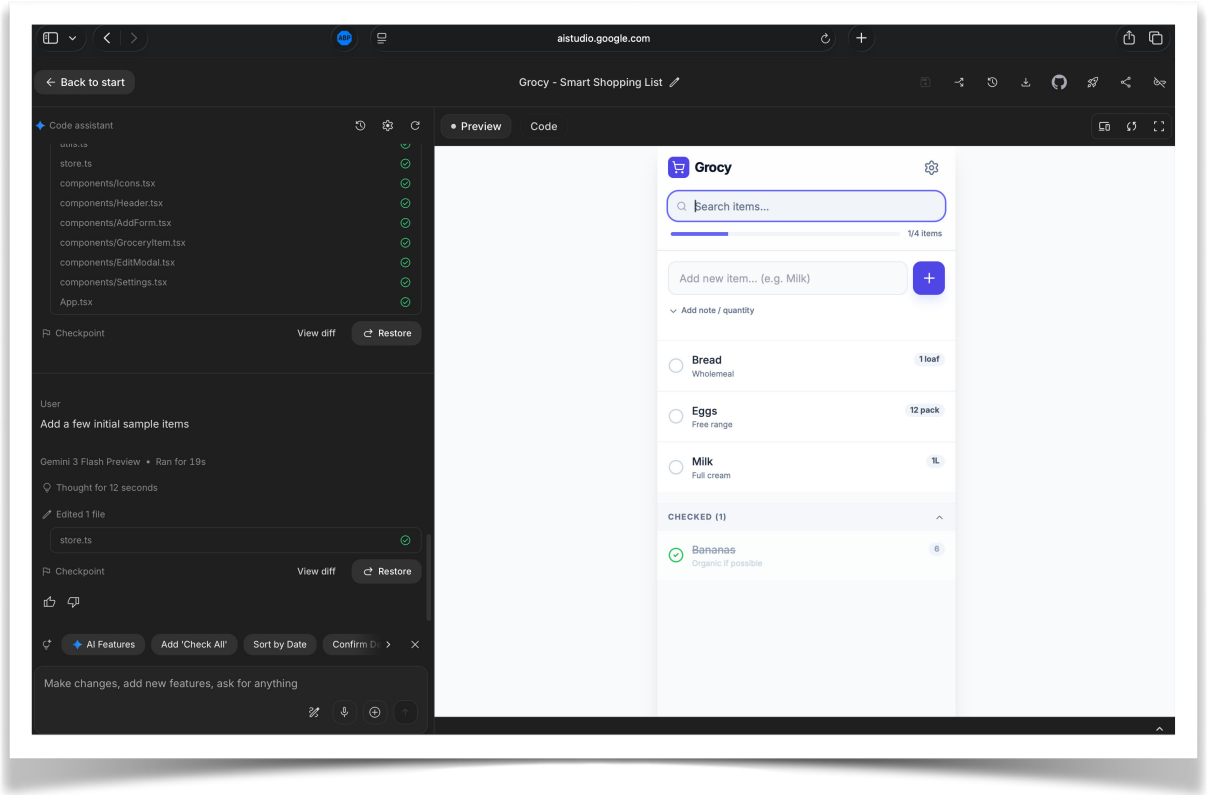
GPT 5.2 က (၄) သိန်းလောက်ထိ လက်ခံပြီး Claude Opus 4.5 က (၂) သိန်းလောက်ထိ လက်ခံနိုင်ပါတယ်။

ဒီပမာဏတွေဟာ မနည်းမနော ဆိုပေမဲ့ ကိုယ်ကပြောလိုက်၊ သူကလုပ်လိုက်၊ ကုဒ်တွေ ရေးလိုက်နဲ့၊ တဖြည်းဖြည်း များလာပြီး ပြည့်သွားတာ မလောက်တာ ဖြစ်တတ်ပါတယ်။ အကယ်၍ လက်ခံနိုင်တဲ့ပမာဏကို ကျော်သွားရင် AI က တချို့ကို ချန်ထားခဲ့မှာ ဖြစ်ပါတယ်။ အဲ့ဒီမှာ အစပိုင်း အဆင်ပြေပေမဲ့ ကြာလာတော့ အဆင်မပြေတော့ဘူး ဆိုတာမျိုးတွေ ဖြစ်တတ်ပါတယ်။ ရှေ့ပိုင်းကတချို့ အချက်အလက်တွေ ချန်ထားခဲ့လို့ Context လွတ်သွားပြီး အဆင်မပြေ ဖြစ်သွားတတ်တာပါ။

လက်ရှိလုပ်နေတဲ့ နမူနာတွေက အဲ့ဒီလောက်ပမာဏကို ကျော်သွားစရာတော့ မရှိသေးပါဘူး။ ထည့်သွင်းသတိပြုစရာအနေနဲ့သာ မှတ်ထားရမှာပါ။

Database

လုပ်လက်စ App ဘက်ကို ပြန်သွားပါမယ်။ စာရေးသူက စောစောကရရှိထားတဲ့ စာမျက်နှာ (၅) မျက်နှာစာလောက်ရှိတဲ့ Detail Prompt ပေးပြီး AI Studio နဲ့ ဖန်တီးလိုက်တဲ့အခါ အခုလို App ကို ရရှိပါတယ်။



တစ်ချက်တည်းနဲ့ တော်တော် အဆင်ပြေတဲ့ ရလဒ်ကို ရလိုက်တာပါ။ စစချင်းရလဒ်မှာ နမူနာ Data တွေ မပါတော့ အလွတ်ကြီးဖြစ်နေလို့ အခုလိုတော့ နောက်ဆက်တွဲ Prompt တိုလေးတစ်ခု ထပ်ပေးလိုက်ရပါတယ်။

Add a few initial sample items

ပထမနမူနာက Static Website စာမျက်နှာလေး တစ်ခုပါပဲ။ အခုဒီနမူနာကတော့ Input တွေပေး အသုံးပြုလို့ရတဲ့ App လေးတစ်ခု ဖြစ်သွားပါပြီ။ ဒါပေမဲ့ ဒါဟာ ကိုယ့် Device ပေါ်မှာ သီးခြားအလုပ်လုပ်တဲ့ Standalone App ဖြစ်ပါတယ်။ တစ်ဦးတည်းပဲ သုံးလို့ရမှာပါ။

ဒါကြောင့် နောက်တစ်ဆင့်အနေနဲ့ ဒီ App ကို အိမ်ကမိသားစုတွေ အတူတူ သုံးလို့ရအောင် လုပ်ကြပါတော့မယ်။ ဒီတော့ System Thinking ပြန်လိုလာပါပြီ။ ထုံးစံအတိုင်း ဘယ်လို ချိတ်ဆက် အလုပ်လုပ်တာ ဖြစ်နိုင်သလဲ ကိုယ့်ခေါင်းထဲမှာ ကြိုရှင်းထားရင် အကောင်းဆုံးပါပဲ။

- အချက်အလက်တွေကို ကိုယ့် Device ထဲမှာသိမ်းလို့ အဆင်မပြေတော့ဘူး၊
- ဗဟိုဆာဗာတစ်ခုမှာသိမ်းပြီး အားလုံးဝိုင်းသုံးလို့ရအောင် လုပ်ရတော့မယ်၊
- အဲ့ဒီလိုသိမ်းလို့ရတဲ့ နည်းပညာ လိုအပ်တယ်၊
- အဲ့ဒီလိုသိမ်းထားတာကို ချိတ်ဆက်တဲ့ နည်းပညာလိုအပ်တယ်၊
- ကိုယ့်မိသားစုမဟုတ်သူတွေ ဝင်သုံးလို့ရနေရင် အဆင်မပြေဘူး၊
- ကိုယ့်မိသားစုဝင်တွေအတွက် အကောင့်တွေလိုတယ်၊

ဘယ်လိုအလုပ်လုပ်သလဲဆိုတဲ့ နည်းပညာ Detail ကို မသိရင်တောင်မှ အနည်းဆုံး ဒီလောက်ကို တွေးကြည့် မြင်ကြည့်နိုင်ဖို့ လိုပါတယ်။ ဒီတော့မှ ကိုယ်က AI နဲ့ တန်းတူတိုင်ပင် အလုပ်လုပ်နိုင်မှာပါ။ မဟုတ်ရင် AI က ဘာတွေလုပ်သွားမှန်း မသိသလို၊ ကိုယ်

ကလည်း အဆင်ပြေတာတွေ မပြေတာတွေမသိ၊ ဘာဆက်ပြောရ လုပ်ရမှန်းမသိတွေ ဖြစ်နေပါလိမ့်မယ်။ နည်းပညာအသေးစိတ်ကို AI ကို ခိုင်းလို့ရပါတယ်။ လိုချင်တဲ့ စနစ်ရဲ့ စီးဆင်းအလုပ်လုပ်သင့်တဲ့ System ကိုတော့ ကိုယ်က မြင်ကြည့်နိုင်ရမှာ ဖြစ်ပါတယ်။

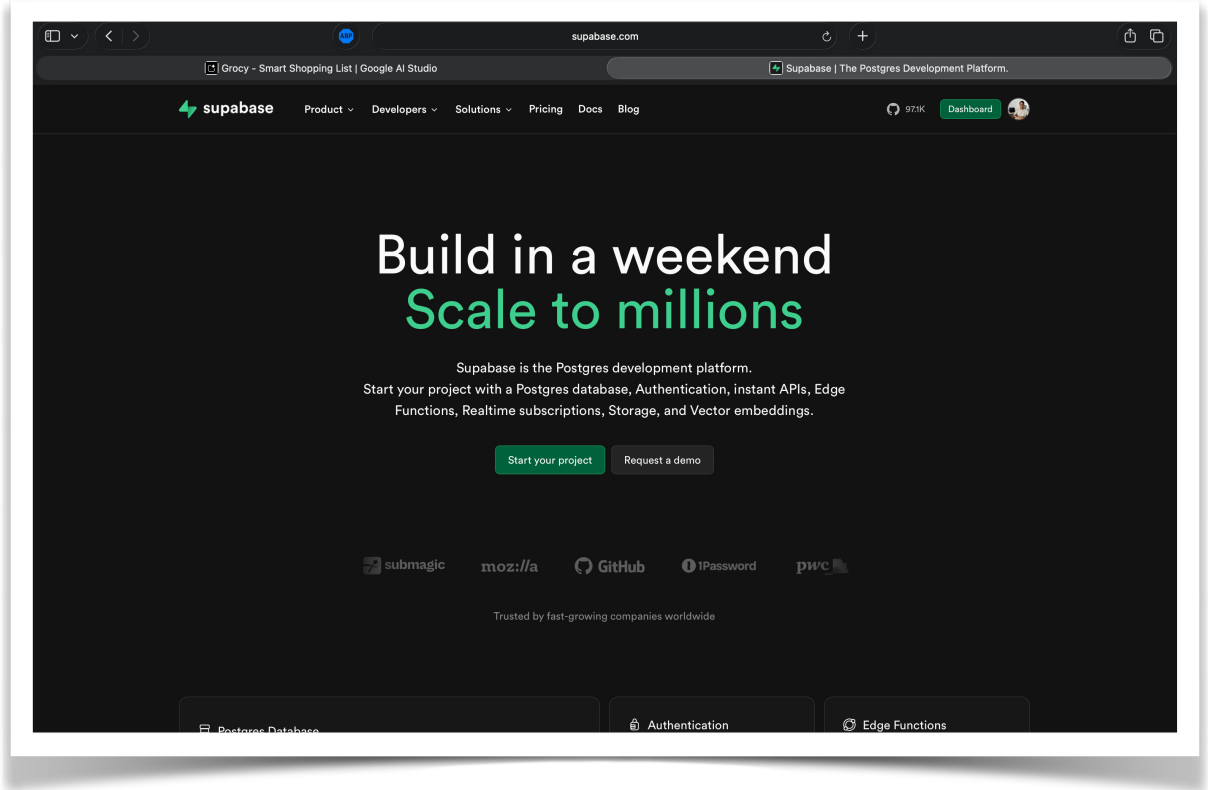
ဒါကြောင့် စာဖတ်သူက ကိုယ်တိုင်ကုဒ်မရေးတတ်ပေမဲ့ အလုပ်အတွေ့အကြုံ တစ်ချို့ရှိလို ဒီလိုစနစ်တွေကို မြင်တတ်တယ် ဆိုတဲ့သူများဖြစ်ခဲ့ရင် တော်တော့်ကို အဆင်ပြေသွားမှာပါ။

စနစ်ကိုမြင်ပြီးနောက် AI ချည်းပဲမရတော့ဘဲ ကိုယ်ကူရမှာ ရှိလာပါပြီ။ ဟိုးရှေ့ဆုံးခန်းမှာ ပြောခဲ့သလို AI ကိုပဲ ကိုယ်နားလည်တဲ့ စနစ်ကို ပြောပြပြီး၊ ငါဘက်က ဘာလုပ်ပေးရမလဲ မေးကြည့်လိုက်လို့ရပါတယ်။ ဒီနေရာမှာတော့ စာရေးသူက ပြောပြသွားမှာပါ။

အောက်ပါလင့်ကို သွားလိုက်ပါ။

https://supabase.com/

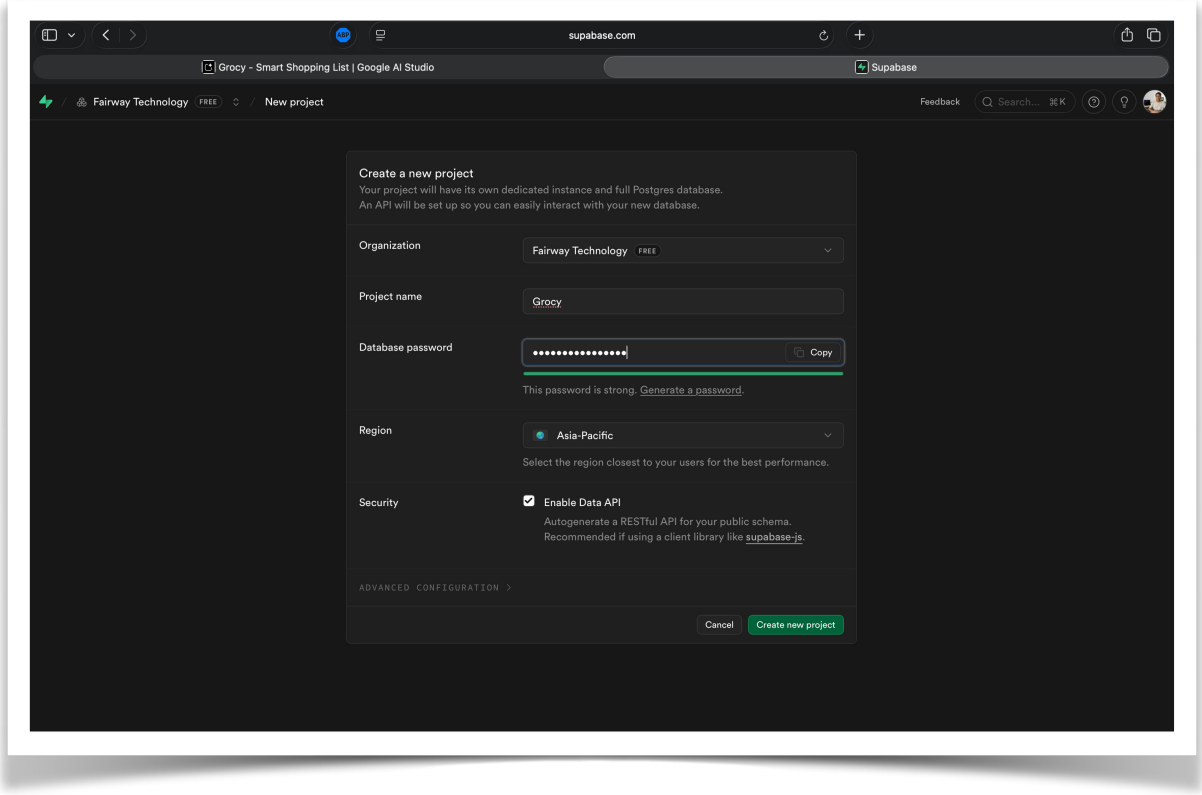
ဒါဟာ Supabase ခေါ်အချက်အလက်ဒေတာတွေ သိမ်းလို့ရတဲ့ ဗဟို Database ဆာဗာ နည်းပညာ ဝန်ဆောင်မှုပါ။ Database ကို ကိုယ်တိုင် စီမံစရာမလိုဘဲ သူက စီမံပေးပါတယ်။ ကိုယ်က ချိတ်ဆက်အသုံးပြုရုံပါပဲ။ Free Version နဲ့ တင် တော်တော်များများသုံးလို့ရပါတယ်။



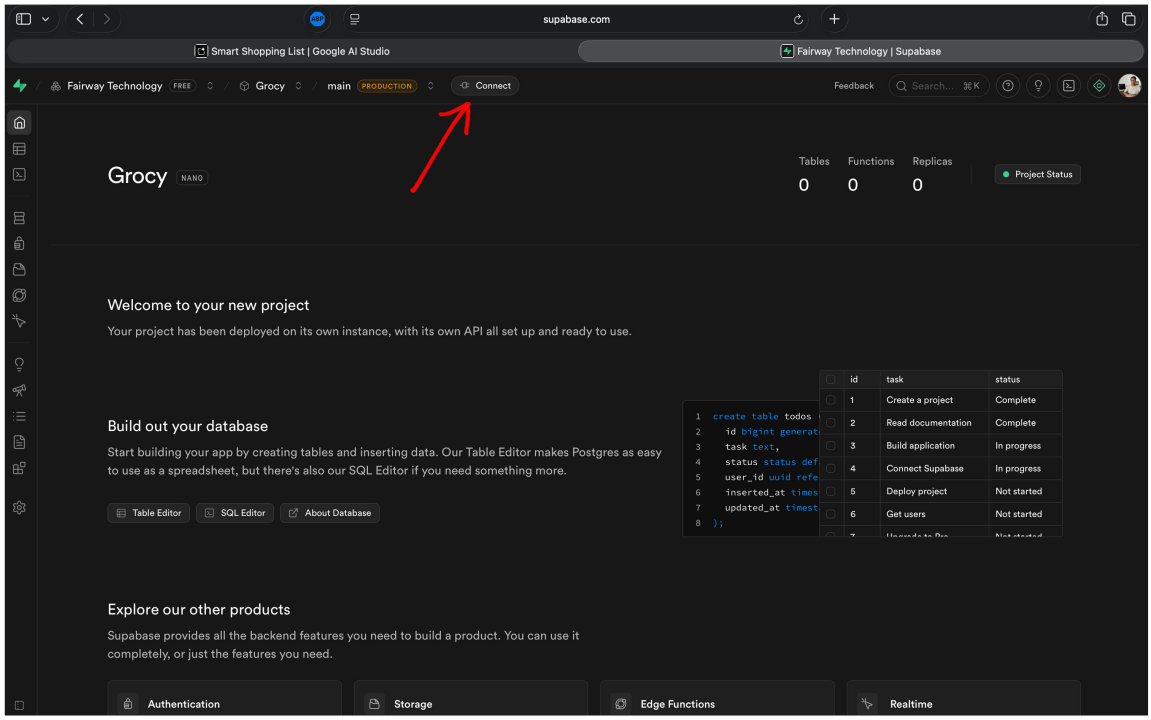
Database တင်မဟုတ်ပါဘူး၊ Authentication ခေါ် User Account စီမံတာတွေ၊ ဖိုင်တွေ စီမံတာတာတွေထိ သူအကုန်လုပ်ပေးပါတယ်။ ဒါကြောင့် ကိုယ်တိုင်ပရိုဂရမ်မာ မဟုတ်သူတွေနဲ့ သင့်တော်တဲ့ နည်းပညာတစ်ခုပါ။ ဒီတစ်ခုတည်း ရှိတာတော့ မဟုတ်ပါဘူး။ အလားတူနည်းပညာတွေ အများကြီး ရှိပါတယ်။ စိတ်ဝင်စားရင် နောက်တော့ ရှာဖွေလေ့လာကြည့်ပါ။

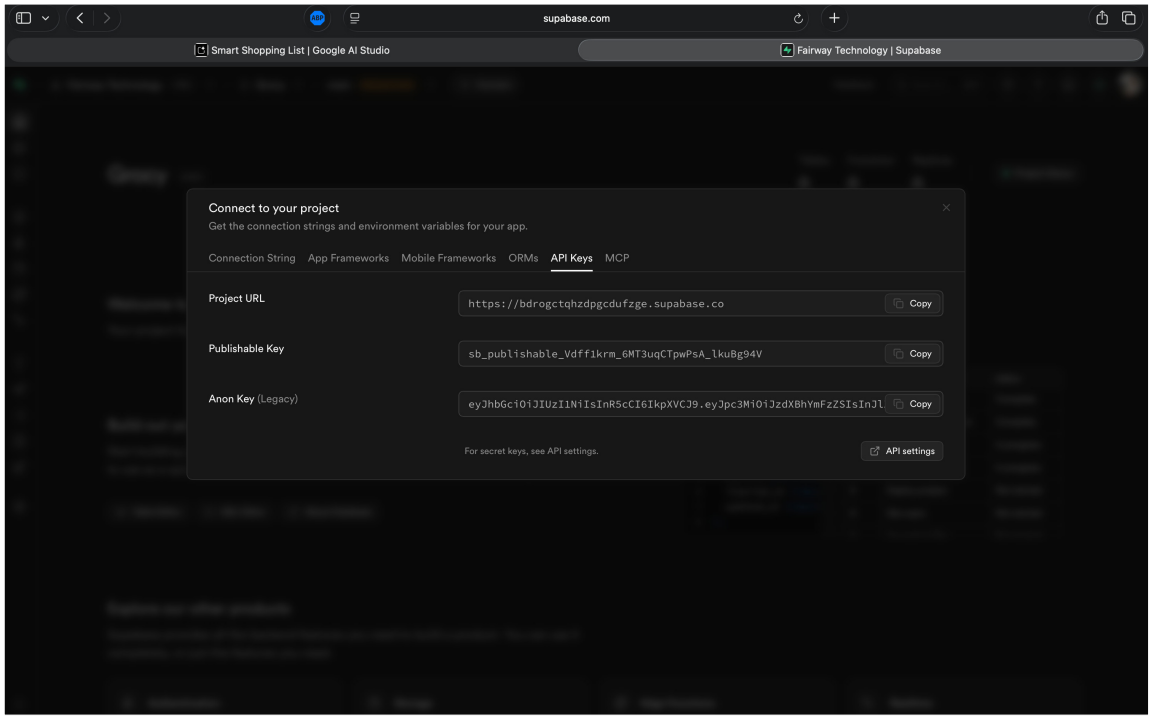
- ပထမဆုံး Supabase အကောင့်ဖွင့်လိုက်ပါ။
- ပြီးတဲ့အခါ Orgranization တစ်ခု ဖန်တီးလိုက်ပါ။ မိမိနှစ်သက်ရာအမည် ပေးနိုင်ပါတယ်။

- ပြီးတဲ့အခါ Project တစ်ခု ဖန်တီးပါ။ ဆက်လက်ဖော်ပြထားတဲ့ နမူနာမှာလိုပဲ လိုအပ်တဲ့အချက်အလက်တွေ ဖောင်မှာဖြည့်ရုံပါပဲ သိပ်မခက်ပါဘူး။



ပြီးတဲ့အခါ အောက်ကပုံမှာ မျှားပြထားတဲ့ Connect ခလုတ်ကို နှိပ်ပြီး API Keys ကို ရွေးပါ။ အဲ့ဒီထဲက **Project URL** နဲ့ **Publishable Key** တို့ကို ကူးယူပါ။


supabase.com
Smart Shopping List | Google AI Studio
Fairway Technology | Supabase
Fairway Technology FREE
Grocy
main PRODUCTION
Connect
Feedback
Search... ⌘K
Grocy NANO
Tables 0
Functions 0
Replicas 0
Project Status
Welcome to your new project
Your project has been deployed on its own instance, with its own API all set up and ready to use.
Build out your database
Start building your app by creating tables and inserting data. Our Table Editor makes Postgres as easy to use as a spreadsheet, but there's also our SQL Editor if you need something more.
Table Editor
SQL Editor
About Database
id task status
1 Create a project Complete
2 Read documentation Complete
3 Build application In progress
4 Connect Supabase In progress
5 Deploy project Not started
6 Get users Not started
Explore our other products
Supabase provides all the backend features you need to build a product. You can use it completely, or just the features you need.
Authentication
Storage
Edge Functions
Realtime


supabase.com
Smart Shopping List | Google AI Studio
Fairway Technology | Supabase
Connect to your project
Get the connection strings and environment variables for your app.
Connection String
App Frameworks
Mobile Frameworks
ORMs
API Keys
MCP
Project URL
https://bdrogctqhzdpgcdufzge.supabase.co
Copy
Publishable Key
sb_publishable_Vdff1krm_6MT3uqCTpwPsA_lkuBg94V
Copy
Anon Key (Legacy)
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJl
Copy
For secret keys, see API settings.
API settings

ပြီးတဲ့အခါ AI Studio မှာ လုပ်လက်စ Grocery App မှာ အခုလို Prompt ပေးလိုက်ပါ။

> *Here are supabase url [project url] and publishable key [publishable_key]. I want to create three user accounts for my family. Everyone should be able to login and add items. Implement it and* ***tell me what I need to do on Supabase****.*

`[project_url]` နဲ့ `[publishable_key]` နေရာမှာ ကိုယ်ရရှိထားတဲ့ အချက်အလက်မှန် ထည့်ပေးဖို့တော့ လိုပါတယ်။ ဒီလိုလေးပြောလိုက်တယ်ဆိုရင် AI Studio က ကိုယ့် ပရောဂျက်ကို Supabase Database နဲ့ ချိတ်ဆက်တဲ့ အလုပ်တွေ လုပ်ပေးသွားမှာပါ။

ဒါပေမဲ့ Database က အလွတ်ပဲ ရှိပါသေးတယ် အထဲမှာ ဘာမှ မရှိသေးပါဘူး။ စောစောက Prompt မှာ “**ငါ Supabase မှာ ဘာလုပ်ပေးရမလဲ**” လို့ ထည့်မေးထားပါတယ်။

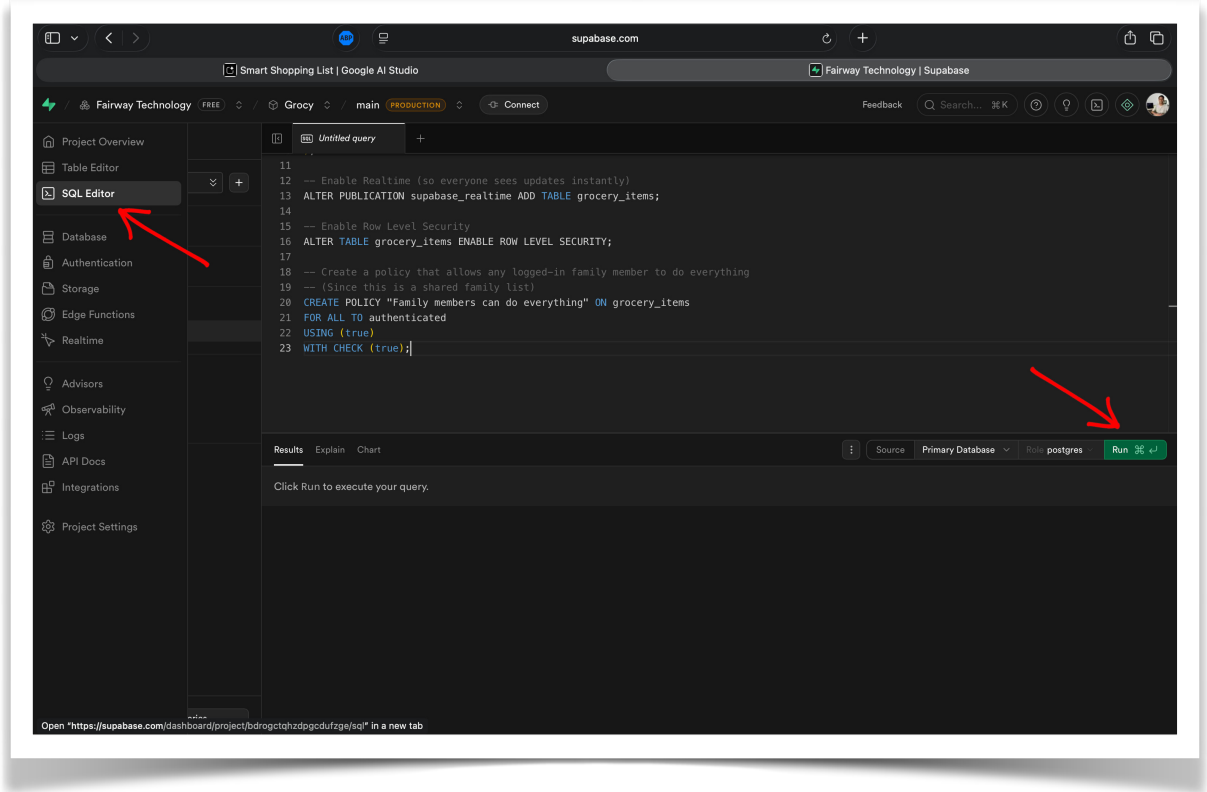
စာရေးသူစမ်းကြည့်တဲ့အခါ AI က ကုဒ်တွေရေးပေးသွားပေမဲ့၊ Supabase မှာ ဘာလုပ်ရမလဲ မပြောလို့ နောက်တစ်ကြိမ် ထပ်မေးလိုက်ရပါတယ်။ အဲ့ဒီလိုမေးလိုက်တဲ့အခါ ဒီကုဒ်ကို Run လိုက်ပါလို့ ပြောလာပါတယ်။

```
-- Create the grocery items table
CREATE TABLE grocery_items (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  name TEXT NOT NULL,
  note TEXT DEFAULT '',
  quantity TEXT DEFAULT '',
  "isChecked" BOOLEAN DEFAULT false,
```

```
  "createdAt" TIMESTAMPTZ DEFAULT now(),
  "updatedAt" TIMESTAMPTZ DEFAULT now()
);

-- Enable Realtime (so everyone sees updates instantly)
ALTER PUBLICATION supabase_realtime ADD TABLE grocery_items;

-- Enable Row Level Security
ALTER TABLE grocery_items ENABLE ROW LEVEL SECURITY;

-- Create a policy that allows any logged-in family member
CREATE POLICY "Family members can do everything" ON grocery_items
FOR ALL TO authenticated
USING (true)
WITH CHECK (true);
```

ဒါကြောင့် နောက်စာမျက်နှာက ပုံမှာပြထားသလို Supabase ရဲ့ SQL Editor မှာ သူပေးတဲ့ကုဒ်ကိုကူးထည့်ပြီး Run လိုက်ပါတယ်။

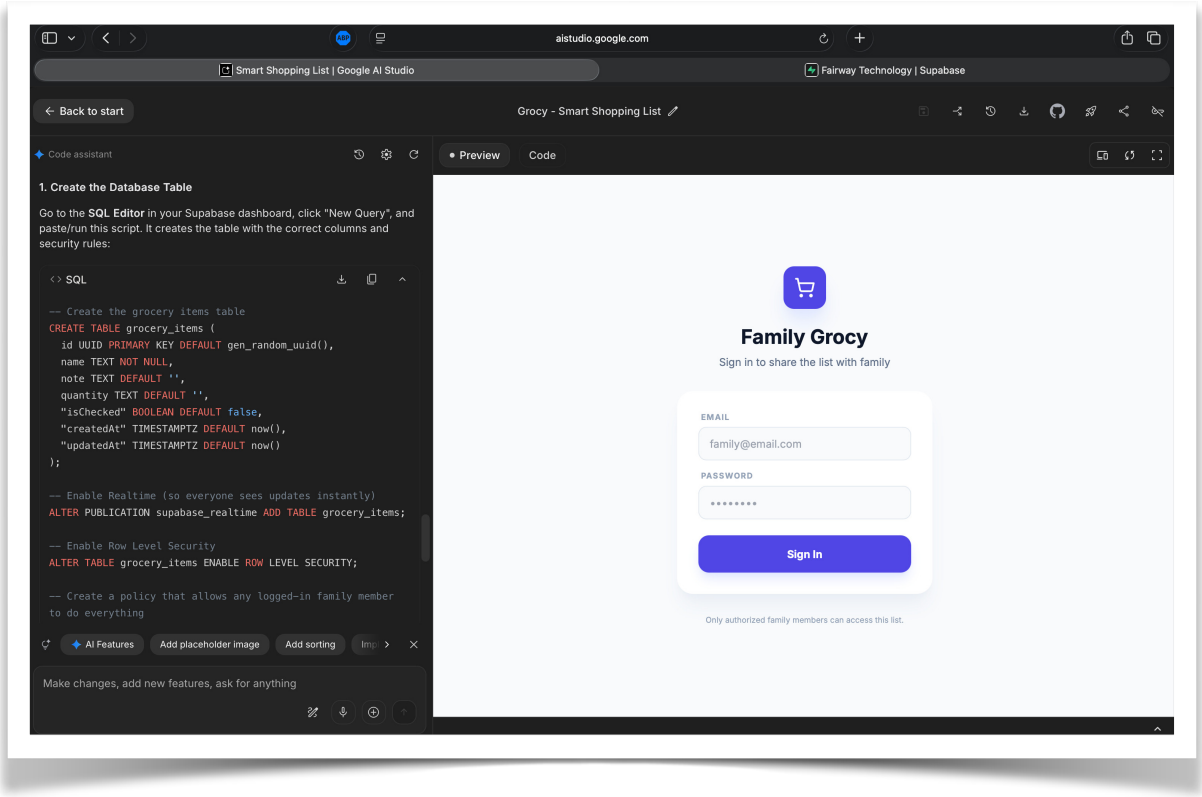
ဒါ ဒေတာတွေ သိမ်းဖို့အတွက် လိုအပ်တဲ့ Table တွေ တည်ဆောက်လိုက်တာပါ။ စာဖတ်သူက ပရိုဂရမ်မာ မဟုတ်လို့ Database နည်းပညာတွေ SQL Code တွေကို မသိရင်တောင် အချက်အလက် ဒေတာတွေကို Column တွေ Row တွေနဲ့ Table ပုံစံသိမ်းရတယ် ဆိုတဲ့သဘောကိုတော့ သိလိမ့်မယ်လို့ ယူဆပါတယ်။



ပေးထားတဲ့ကုဒ်အရ `grocery_items` အမည်နဲ့ Table တစ်ခု တည်ဆောက်ပြီး `id`, `name`, `note` နဲ့ `quantity` တို့လို သိမ်းချင်တဲ့ အချက်အလက်တွေအတွက် Column တွေ ထည့်ပေးလိုက်တာပါ။

အရေးကြီးဆုံး တစ်ချက်ကတော့ Row Level Security ဆိုတဲ့ လုပ်ဆောင်ချက်ပါပဲ။ ကုဒ်တွေ မသိရင်တောင် ဒီလိုလေး မှတ်ထားပေးပါ။ Default အနေနဲ့ Supabase မှာ ဒေတာတွေကို မည်သူမဆို ဝင်ကြည့်လို့ရပါတယ်။ အဲ့ဒီလို ကြည့်လို့ မရစေချင်ရင် Row Level Security ထည့်ပေးရပါတယ်။ AI ပေးတဲ့ ကုဒ်ထဲမှာ Security ထည့်ပေးတဲ့ ကုဒ်လည်း ပါပါတယ်။ ဒါကြောင့် အကောင့်နဲ့ Login ဝင်ထားမှာပဲ ကြည့်လို့ရတော့မှာပါ။

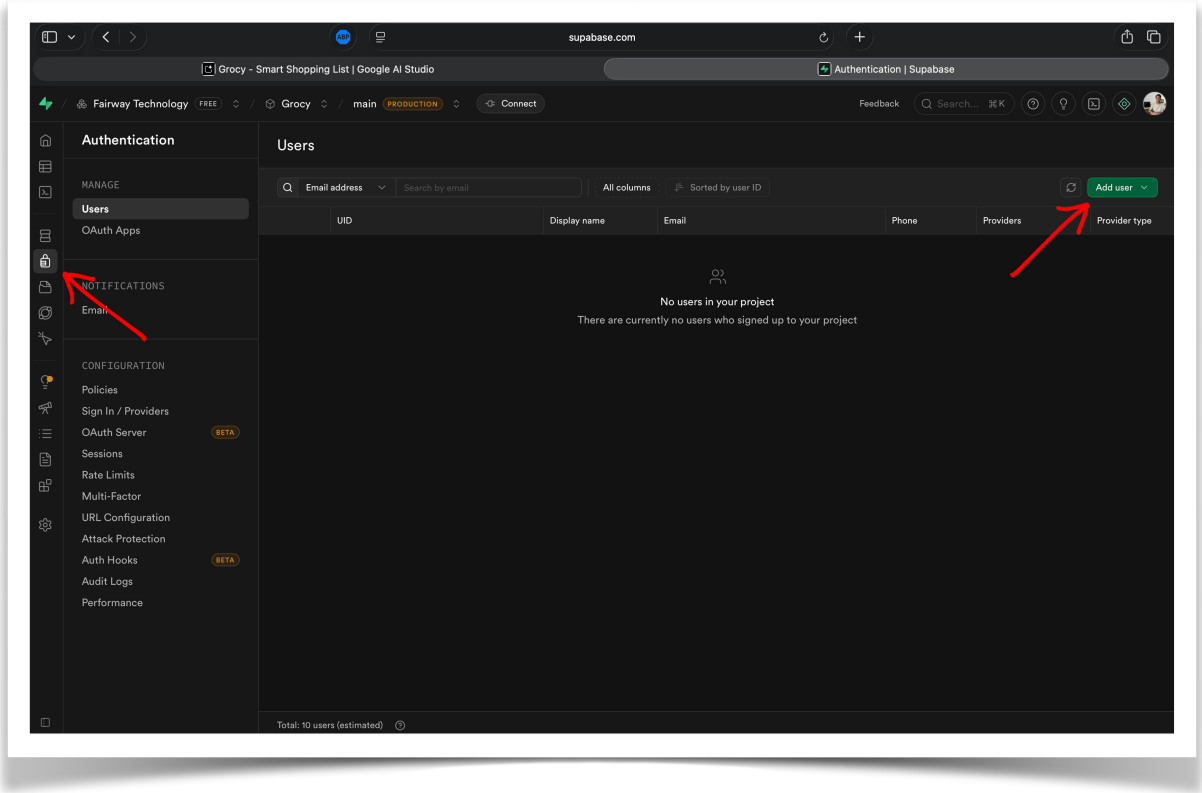
နောက်ပိုင်းမှာ Supabase ကို ဆက်လက်အသုံးပြုဖြစ်ရင် အဲ့ဒီအချက်ကို ကြပ်ကြပ်လေး သတိထားဖို့ လိုပါလိမ့်မယ်။ Security မထည့်မချင်း လူတိုင်း ယူလို့ရနေမှာဖြစ်ပြီး Security ထည့်လိုက်မှသာ ဘယ်သူပဲ ယူလိုရမယ်ဆိုတဲ့ ကန့်သတ်ချက် ပါသွားတာပါ။



နမူနာမှာကြည့်လိုက်ရင် အကောင့်နဲ့ Login ဝင်ပြီး အချက်အလက်တွေ အသုံးပြုလို့ ရအောင် လုပ်ပေးထားပြီး ဖြစ်နေတာကို တွေ့ရပါတယ်။

စာရေးသူ Login ဝင်မလို့ စမ်းကြည့်တဲ့အခါ ရိုက်ထည့်တဲ့အခါ စာလုံးတွေ မမြင်ရဘူး ဖြစ်နေလို့ အဲ့ဒါလေး ပြောပြပြီး ပြင်ခိုင်းလိုက်ရပါတယ်။ ဒီလိုပါပဲ။ အမှားလေးတွေ ကျန်တတ်ပါတယ်။ အဲ့ဒါလေးတွေ AI ကိုပဲ ပြောပြီး ပြင်ခိုင်းရမှာပါ။

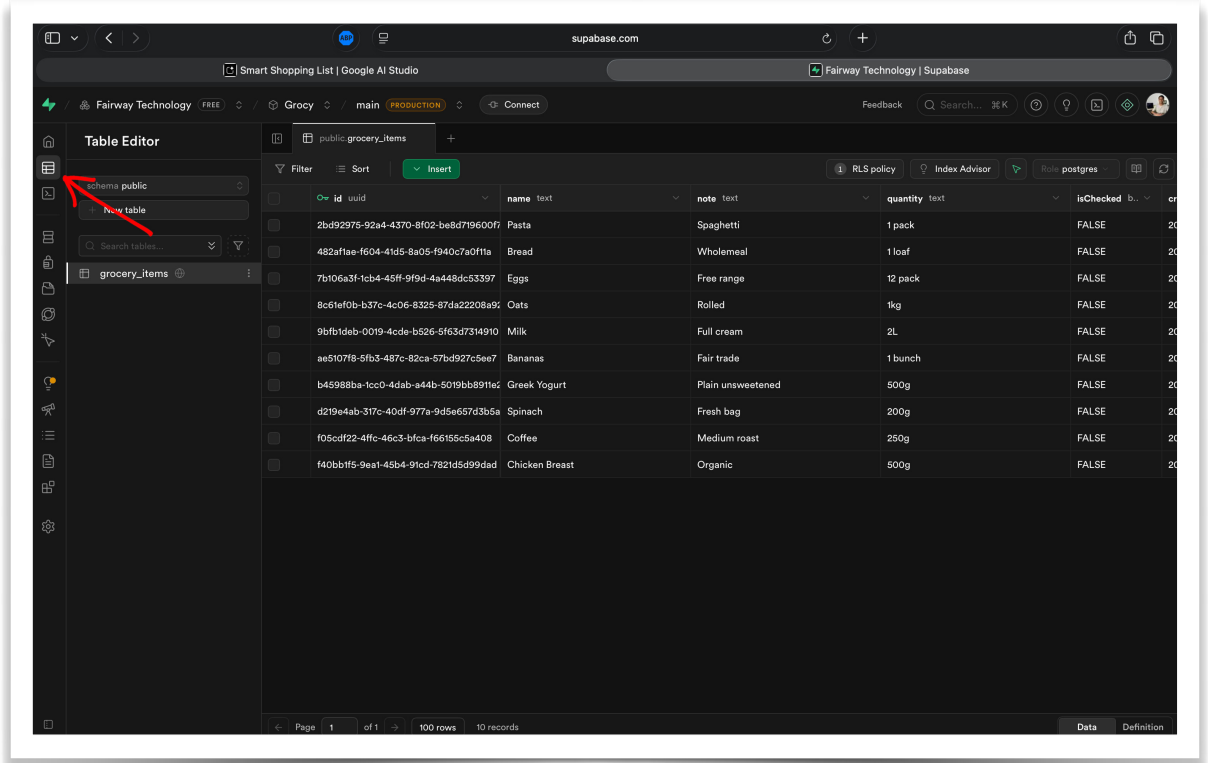
Login တကယ်ဝင်လို့ မရသေးပါဘူး။ အကောင့်တွေ မရှိသေးလို့ပါ။ ဒါကြောင့် နောက်တစ်ဆင့်အနေနဲ့ Login ဝင်လို့ရအောင် အကောင့်တွေ တည်ဆောက်ပါမယ်။ ဆက်လက်ဖော်ပြထားတဲ့ ပုံမှာ မျှားထိုးပြထားသလို Supabase ရဲ့ Authentication ကို သွားလိုက်ပါ။ ပြီးတဲ့အခါ **Add user** ကို နှိပ်ပြီး User တွေ ကြိုထည့်ထားလိုက်ပါ။



အကောင့်တွေ ထည့်ပြီးနောက် Login ဝင်လိုက်တဲ့အခါ ရသွားပါတယ်။ ဒါပေမဲ့ လိုအပ်တဲ့ Database တွေ Table တွေ အထဲမှာ ကြိုထည့်ထားတဲ့ ဒေတာတွေ မရှိလို့ အလွတ်ကြီး ဖြစ်နေပါတယ်။ ဒါကြောင့် **Sample Data တချို့ ထည့်ချင်တယ်၊ ဘယ်လိုလုပ်ရမလဲ** လို့ ထပ်ပြောလိုက်တဲ့အခါ AI က ဒီကုဒ်လေး ထုတ်ပေးပါတယ်။

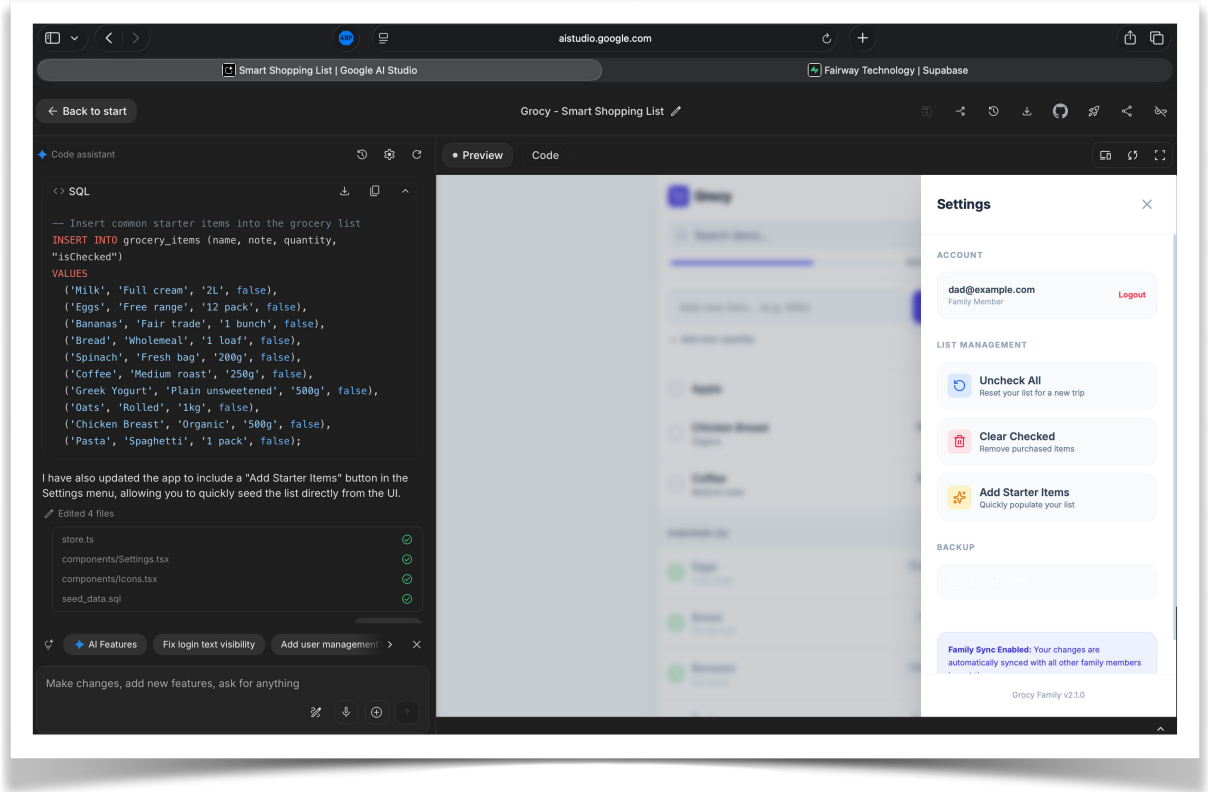
```
-- Insert common starter items into the grocery list
INSERT INTO grocery_items (name, note, quantity, "isChecked")
VALUES
  ('Milk', 'Full cream', '2L', false),
  ('Eggs', 'Free range', '12 pack', false),
  ('Bananas', 'Fair trade', '1 bunch', false),
  ('Bread', 'Wholemeal', '1 loaf', false),
  ('Spinach', 'Fresh bag', '200g', false),
  ('Coffee', 'Medium roast', '250g', false),
  ('Greek Yogurt', 'Plain unsweetened', '500g', false),
  ('Oats', 'Rolled', '1kg', false),
  ('Chicken Breast', 'Organic', '500g', false),
  ('Pasta', 'Spaghetti', '1 pack', false);
```

စောစောကလိုပဲ SQL Editor မှာ ထည့်ပြီး Run ပေးလိုက်တဲ့အခါ စမ်းကြည့်စရာ နမူနာ ဒေတာတချို့ ရရှိသွားပါတယ်။ ဆန္ဒရှိရင် ဒေတာတွေ ကိုယ့်ဘာသာ ထပ်ထည့်တာတွေ ပြင်တာတွေ Supabase ဘက်မှာပဲ လုပ်လို့ရပါတယ်။ ပုံမှာ မျှားပြထားတဲ့ နေရာကို နှိပ်ပြီး စမ်းကြည့်လိုက်ပါ။



အားလုံး ပြည့်စုံသွားပါပြီ။ ကိုယ့် Device ထဲမှာတင် သီးခြားအလုပ်လုပ်တဲ့ Standalone App မဟုတ်တော့ပါဘူး။ Account ရှိတဲ့သူတွေ ဝင်သုံးလို့ရတဲ့ Backend Database တွေ ဘာတွေ ပါတဲ့ App လေးတစ်ခုကို ရရှိသွားပြီ ဖြစ်ပါတယ်။

နောက်ဆုံးရလဒ်အနေနဲ့ အခုလို နမူနာဒေတာတွေလည်းပါတယ်။ အကောင့်နဲ့ Login ဝင်ပြီးတော့လည်း အသုံးပြုလို့ရနေပြီ ဖြစ်ပါတယ်။ နမူနာဒေတာတွေကို Supabase ဘက်မှာ သွားမထည့်ဘဲ ဒီလက်ကနေ ခလုတ်နှိပ်ထည့်လို့ရအောင် **Add Starter Item** ဆိုတဲ့ ခလုတ်လေးပါ ကိုယ်မထည့်ခိုင်းပေမယ့် Settings ထဲမှာ ထည့်ပေးထားတာကိုလည်း တွေ့ရပါတယ်။



အခုဖော်ပြခဲ့တဲ့ နည်းတွေကို ကိုယ့်စိတ်ကူးအိုင်ဒီယာတွေနဲ့ ပေါင်းစပ် ဖန်တီးလိုက်ရင် လုပ်လို့ရတာတွေမှ အများကြီးဖြစ်သွားပါပြီ။ အကန့်အသတ်ဆိုလို့ ကိုယ့်အိုင်ဒီယာပဲ ရှိပါတော့မယ်။ နည်းပညာဘက်ကတော့ တော်တော်ကို လုပ်လို့ရသွားတာပါ။

Database ဆိုတာ အလွန်အန္တရာယ်များပါတယ်။ လုံခြုံရေးသာမက အမှားလုပ်မိရင် အချက်အလက် မတော်တဆ ဆုံးရှုံးတာတွေလည်း ရှိနိုင်ပါတယ်။ **အခန်း (၉)** မှာ ဖော်ပြထားတဲ့ **သတိပြုရန်အချက်များ** ကို သေချာလေ့လာဖို့ လိုပါတယ်။

စာဖတ်သူက ဒီအကြောင်းတွေ သေချာနားမလည်သေးသူဆိုရင် အခုပြတဲ့ နည်းတွေကို စမ်းသပ်တဲ့ သဘောသာ အသုံးပြုသင့်ပါတယ်။ လက်တွေ့အသုံးချဖို့ထိ ရည်ရွယ်တယ် ဆိုရင်တော့ကျွမ်းကျင်သူနဲ့ ပြပြီး အကြံဉာဏ် စစ်ဆေးမှု ယူသင့်တယ် ဆိုတာလေး သတိပြုပါ။

အခန်း (၈) မှာ အခုလိုဖန်တီးထားတဲ့ App တွေကို အများသုံးလို့ရအောင် ဘယ်လို Publish လုပ်ရလဲထည့်သွင်းဖော်ပြမှာပါ။ အခုတော့ နောက်အဆင့်တွေ ဆက်မသွားခင် သိထားသင့်တဲ့ Coding ပိုင်း ဗဟုသုတတချို့ကို ထည့်ကြည့်ကြပါမယ်။

အခန်း (၅) - Coding ပိုင်းဗဟုသုတအချို့

ဒီအခန်း Coding ပိုင်းဗဟုသုတတွေ ပြောတဲ့အခါ၊ ကုဒ်ရေးနည်းတွေ ဖတ်နည်းတွေ ပြောမှာ မဟုတ်ပါဘူး။ အဲ့ဒီအကြောင်းအရာတွေ စိတ်ဝင်စားရင် စာရေးသူကိုယ်တိုင် ထုတ်ဝေဖူးတဲ့ စာအုပ်တွေ ရှိနေပါတယ်။ အောက်ပါလိပ်စာမှာ ရယူနိုင်ပါတယ်။

https://eimaung.com

ဒီအခန်းမှာတော့ ဘယ်လို Coding နည်းပညာတွေကို ဘယ်လိုနေရာမှာသုံးလဲ၊ ဘာကြောင့်လဲ၊ ဘယ်လို Run ရလဲ၊ ဘယ်လိုအသုံးချရလဲ ဆိုတာတွေကို ပြောပြသွားမှာပါ။ ဆိုလိုတာက အင်ဂျင်တစ်လုံးရဲ့ အလုပ်လုပ်ပုံကို မသိသေးရင် ခဏထား၊ စက်နှိုးချင်ရင် ဆီထည့်ပြီး ခလုတ်နှိပ်ရတယ်ဆိုတာမျိုးလောက်တော့ သိရမယ်ဆိုတဲ့သဘောပါ။ အဲ့ဒါတွေ ပြောပြမှာပါ။

ဆော့ဖ်ဝဲအမျိုးအစား

System Software တို့ Network Software တို့လို အခြေခံကျတဲ့ Software တွေကနေ၊ လုပ်ငန်းသုံး Software တွေ၊ Game တွေ၊ App တွေ၊ ဝက်ဘ်ဆိုက်တွေထိ ဆော့ဖ်ဝဲအမျိုးအစား အမျိုးမျိုးကို Vibe Coding နဲ့ ဖန်တီးလို့ရပါတယ်။

ဒါပေမဲ့ Vibe Coding လုပ်ဖို့အတွက် အထူးသင့်တော်တဲ့ Software တွေက ဒါတွေပါ။

- Web App
- Mobile App
- Backend

ကျန်ဆော့ဖ်ဝဲအမျိုးအစားတွေကို အစပိုင်းမှာ မလုပ်သင့်သလို၊ လုပ်ဖို့ကြိုးစားရင်လည်း မရဖို့များပါတယ်။ ခြွင်းချက်တချို့လောက်ပဲ ရှိပါမယ်။ ဥပမာ ဂိမ်းသေးသေးလေးတွေ လုပ်လို့ရတယ်။ Windows Software အသေးစားတချို့ လုပ်လို့ရတယ် စသည်ဖြင့်။

အမှန်တကယ်တော့ အထက်မှာပြောထားတဲ့ Software အမျိုးအစား (၃) မျိုးနဲ့တင် တော်တော်အဆင်ပြေနေပါပြီ။ ဒီသုံးမျိုးက သီးခြားစီလည်း မဟုတ်ကြပါဘူး။ အများအားဖြင့် ပရောဂျက်တစ်ခုမှာ သုံးမျိုးလုံးပါအောင် စုပေါင်းဖန်တီးရလေ့ ရှိပါတယ်။

Backend တစ်ခုရှိပြီး အဲ့ဒီ Backend ကို Web Frontend နဲ့ Mobile Frontend တို့က ချိတ်ဆက်အလုပ်လုပ်ကြတာပါ။

Backend ဆိုတဲ့နေရာမှာ ပြီးခဲ့တဲ့အခန်းမှာ လုပ်ခဲ့သလို Database Backend ဖြစ်နိုင်သလို ကိုယ်တိုင်ရေးသားစီမံတဲ့ သီးခြားစနစ်လည်း ဖြစ်နိုင်ပါတယ်။

Vibe Coding လုပ်လိုသူတွေအနေနဲ့ Supabase လို Database Backend Service တွေကိုပဲ Backend အနေနဲ့ အသုံးပြုသင့်ပါတယ်။ ဘာဖြစ်လို့လဲဆိုတော့ လက်တွေ့ပရောဂျက်တွေမှာ ရေးရတာ မခက်ပါဘူး။ AI က ရေးပေးပါတယ်။ အဲ့ဒီ Backend တွေ Database တွေကို စီမံရတာကသာ တော်တော်ခက်တာပါ။ ကိုယ်တိုင်စီမံလိုရင် အများကြီး မြှုပ်နှံလေ့လာထားမှပဲရမှာပါ။

ဒါကြောင့် နောက်ပိုင်းတော့ ဆက်လေ့လာပါ။ အစပိုင်းမှာ Database Backend Service တွေကိုပဲ အားထားပါလို့ ပြောချင်ပါတယ်။

ကျန်နေတဲ့ နှစ်ခုဖြစ်တဲ့ Web နဲ့ Mobile အတွက် အသင့်တော်ဆုံးနည်းပညာကတော့ JavaScript ပါပဲ။

တခြားနည်းပညာတွေလည်း ရှိကြပါတယ်။ ကြိုက်ရင်သုံးလို့ရပါတယ်။ ဥပမာ - Python တို့ Flutter တို့လို နည်းပညာတွေကို လူတွေ စိတ်ဝင်စားကြပါတယ်။ ဒါပေမဲ့ ပြဿနာက Flutter နဲ့ Mobile ပဲ လုပ်လို့ရပါတယ်။ Web အတွက် အဆင်ပြေမှာ မဟုတ်ပါဘူး။ Python နဲ့ Backend ပဲ ရပါမယ်။ Frontend နှစ်မျိုးလုံးကို မရနိုင်ပါဘူး။

ဒါကြောင့် အဓိကအသုံးပြုရမယ့် နည်းပညာအုပ်စု တစ်စုတည်းပဲ ရှိပါတယ်။ JavaScript နဲ့ JavaScript ပေါ်မှာ အခြေခံထားတဲ့ ဆက်စပ်နည်းပညာတွေပါ။

JavaScript Ecosystem

App အပါအဝင် ဆော့ဖ်ဝဲတွေဖန်တီးဖို့အတွက် နည်းပညာ အလွှာပေါင်းများစွာ လိုပါတယ်။ ကုဒ်ရေးလိုက်ရုံနဲ့ အဲ့ဒီကုဒ်ကနေ ဆော့ဖ်ဝဲဖြစ်သွားတာမျိုး မဟုတ်ပါဘူး။ JavaScript Ecosystem ထဲမှာပါတဲ့ အလွှာအားလုံးကို တစ်ထိုင်တည်းအကုန်ပါအောင် ဖော်ပြဖို့ မလွယ်ပေမဲ့ အဓိကအကျဆုံးနည်းပညာတွေက ဒါတွေပါ။

- JavaScript
- HTML/CSS
- TypeScript
- Node
- NPM
- Vite
- React
- React Native
- Expo
- Nextjs
- Express

မနည်းမနောပါပဲ။ အားလုံးရဲ့အကြောင်းကို အကျဉ်းချုပ်ပြောပြပါမယ်။ **JavaScript** ဟာ ပင်မ Programming Language ဖြစ်ပါတယ်။ ကျန်အလွှာတွေ မပါသေးဘဲ သူ့ချည်းပဲ အသုံးပြုပြီး App တွေ လုပ်မယ်ဆိုရင် ရနိုင်ပါတယ်။

ရနိုင်တယ်ဆိုပေမဲ့ မပြည့်စုံပါဘူး။ UI အသွင်အပြင်တွေ ဖော်ပြနိုင်ဖို့အတွက် **HTML/CSS** နဲ့ ပူးတွဲ အသုံးပြုရပါတယ်။ အားလုံး နေ့စဉ်အသုံးပြုနေတဲ့ Google Chrome, Microsoft Edge, Mozilla Firefox စတဲ့ Web Browser တွေဟာ HTML/CSS နဲ့ JavaScript ကုဒ်တွေကို Run ပေးနိုင်တဲ့ နည်းပညာ ဖြစ်ပါတယ်။ ဒါကြောင့် HTML/CSS/JavaScript နဲ့ App တွေဖန်တီးလိုက်ရင် နှစ်သက်ရာ Browser နဲ့ အသုံးပြုလို့ရသွားတာပါ။

JavaScript ဟာ ရေးသားရလွယ်ကူတဲ့ Language တမျိုးပါ။ အဲ့ဒီလို ရေးသားရလွယ်ကူတဲ့ သဘောကြောင့်ပဲ မတော်တဆ မှားနိုင်တာတွေလည်း ပိုများပါတယ်။ ဒါကြောင့် **TypeScript** ဆိုတဲ့ နောက်ထပ် Language ကို ထပ်ဆင့်တီထွင် အသုံးပြုကြပါတယ်။

TypeScript မှာ အနည်းငယ်ပိုမိုတင်းကြပ်သွားတဲ့ Language ဖြစ်ပြီး JavaScript ထက် မတော်တဆလုပ်မိတဲ့ အမှားတွေကို နည်းစေပါတယ်။ ဒါကြောင့် နောက်ပိုင်းမှာ JavaScript ကို တိုက်ရိုက်အသုံးမပြုကြဘဲ TypeScript ကို တစ်ဆင့်ခံ အသုံးပြုတာ များလာပါတယ်။

ဒါပေမဲ့ ပြဿနာက TypeScript ကုဒ်တွေကို တိုက်ရိုက် Run လို့မရပါဘူး။ JavaScript ပြောင်းပြီး Run ရပါတယ်။ ဒါကြောင့် ကြားခံပြောင်းပေးနိုင်တဲ့ နည်းပညာတွေလည်း လိုအပ်လာပါတယ်။

Node ကတော့ JavaScript ကုဒ်တွေကို Run ပေးနိုင်တဲ့ Runtime နည်းပညာပါ။ ဒါကြောင့် JavaScript ကုဒ်တွေကို Web Browser ထဲမှာ Run လို့ ရသလို Node နဲ့လည်း

Run လို့ရတယ် လို့ အကြမ်းဖျဉ်း ပြောနိုင်ပါတယ်။ Node လိုမျိုး JavaScript တွေ Run ပေးနိုင်တဲ့ Deno တို့ Bun တို့လို့ နည်းပညာ မူကွဲတွေ ရှိကြပါသေးတယ်။

NPM ဆိုတာ ကိုယ့်ပရောဂျက်မှာ လိုအပ်တဲ့ JavaScript Package တွေကို Download ရယူပေးနိုင်တဲ့ နည်းပညာပါ။ အရာရာကိုယ်တိုင်ရေးစရာ မလိုပါဘူး။ အင်တာနက်မှာ အဆင်သင့် ကြိုရေးထားပြီးသား၊ ရယူအသုံးပြုလို့ရတဲ့ Package သန်းပေါင်းများစွာ ရှိနေပါတယ်။ အဲ့ဒီ Package တွေကို ကိုယ့်ပရောဂျက်မှာ ထည့်သုံးချင်ရင် NPM နဲ့ ရယူရတာပါ။ Yarn တို့ pnpm တို့လို အလားတူ နည်းပညာမူကွဲတွေ ရှိကြပါသေးတယ်။

Vite ဟာ Build System တမျိုးပါ။ ဟိုးအပေါ်မှာ အကြမ်းဖျဉ်းပြောခဲ့သလိုပဲ ကုဒ်တွေ ရေးပြီးနောက် Run လို့ရအောင် Build လုပ်ပေးရတယ် လို့ ပြောထားပါတယ်။ တခြားမူကွဲတွေ အများကြီးရှိပေမဲ့ Vite ဟာ လက်ရှိ အဲ့ဒီလို Build လုပ်ဖို့အတွက် အသုံးအများဆုံး နည်းပညာတစ်ခုပါ။

React ကတော့ လူသုံးအများဆုံး Frontend Library ဖြစ်ပါတယ်။ လိုရင်းအချုပ်အနေနဲ့ React ကိုသုံးပြီး ကိုယ့်ပရောဂျက်အတွက် UI Component တွေ ဖန်တီးနိုင်သလို၊ React နဲ့ ကြိုတင်ဖန်တီးထားတဲ့ UI Component ပေါင်းများစွာလည်း အသင့်ရှိနေလို့ ရယူအသုံးပြုလို့ ရပါတယ်။

React Native ကတော့ React နဲ့ အလုပ်လုပ်ပုံဆင်တူပြီး Mobile App တွေရေးဖို့ အတွက် အသုံးပြုပါတယ်။ React Native နဲ့ရေးထားတဲ့ App ကို iOS App အဖြစ်ပြောင်းလို့ရသလို၊ Android App အဖြစ်လည်း ပြောင်းလို့ရပါတယ်။ ဒါကြောင့် App အမျိုး

အစား နှစ်မျိုးရဖို့ နှစ်ခါဖန်တီးစရာမလိုဘဲ တစ်ကြိမ်ဖန်တီးလိုက်ရုံနဲ့ နှစ်မျိုးရနိုင်သွားပါတယ်။

Expo ဆိုတာ React Native ပေါ်မှာ ထပ်ဆင့်ဖန်တီးထားတဲ့ Framework ဖြစ်ပါတယ်။ Mobile App တွေ လိုအပ်နိုင်တဲ့ အခြေခံလုပ်ဆောင်ချက်တွေကို ကြိုတင် စုစည်းပေးထားပါတယ်။ ဒါကြောင့် React Native ချည်းမသုံးဘဲ Expo နဲ့ တွဲသုံးရင် လိုအပ်လေ့ရှိတာတွေ တော်တော်များများကို အဆင်သင့်ရသွားမှာပါ။

Nextjs ကတော့ React ပေါ်မှာ ထပ်ဆင့်ဖန်တီးထားတဲ့ Framework ပါ။ အလုပ်လုပ်တဲ့ ပုံစံ နည်းနည်းကွာသွားပါတယ်။ React သက်သက်ဆိုရင် Frontend ဖန်တီးဖို့လို့ ပြောလို့ရပါတယ်။ Nextjs နဲ့ Frontend ရော Backend ပါဖန်တီးမယ်ဆိုရင် ရနိုင်ပါတယ်။ ဒါကြောင့် တချို့ပရောဂျက် အမျိုးအစားတွေနဲ့ ပိုသင့်တော်နိုင်ပါတယ်။

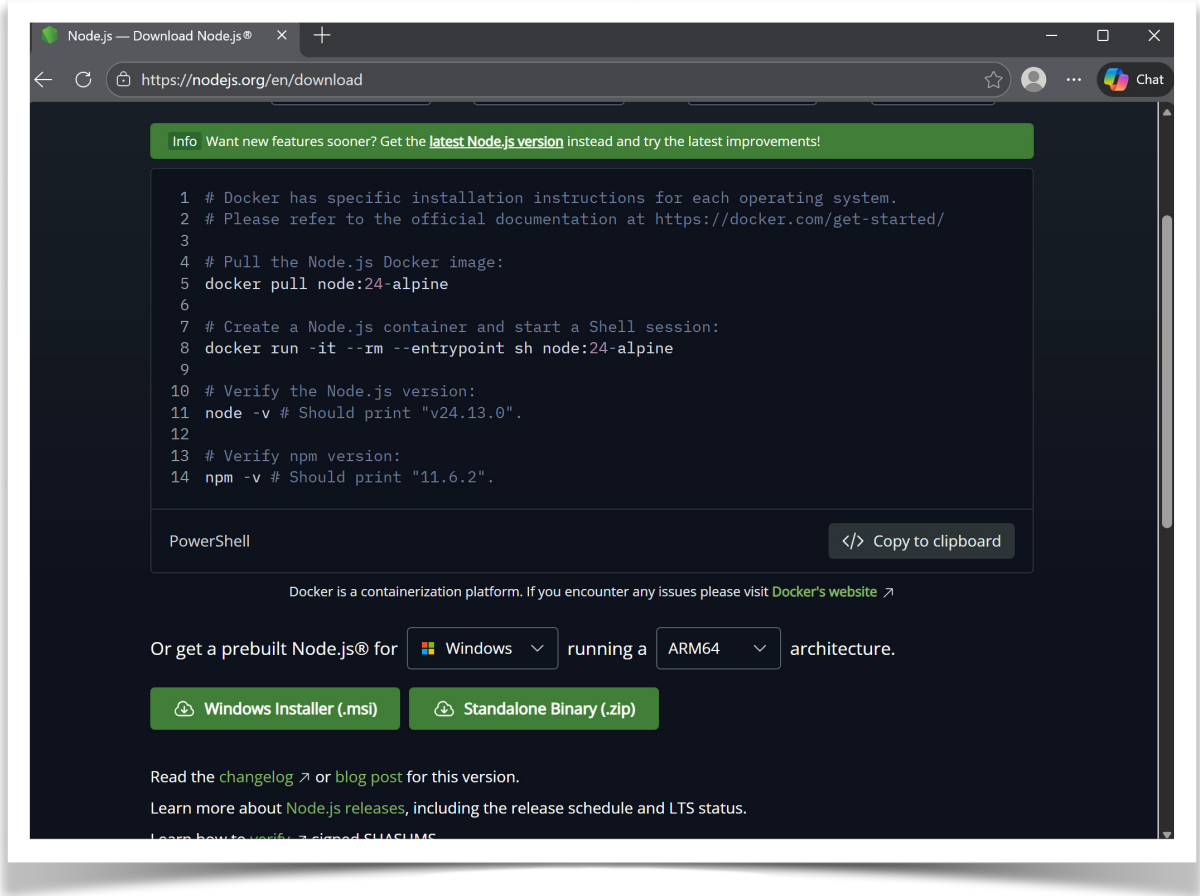
Express ကတော့ Backend တွေရေးဖို့အတွက် သင့်တော်တဲ့ နည်းပညာပါ။ ဒါပေမဲ့ စောစောကပြောခဲ့သလိုပဲ အစပိုင်းမှာ ကိုယ်တိုင်ရေးသားမယ့်အစား Backend Service တွေကိုပဲ ပိုအားပြုစေချင်ပါတယ်။

Building and Running

ဒီနည်းပညာတွေ လက်တွေ့အသုံးချဖို့ တခြားဘာမှ ရှုပ်ရှုပ်ထွေးထွေး မလိုပါဘူး။ Node ကို ကိုယ့်စက်ထဲမှာ Install လုပ်ထားလိုက်ရင် ရပါပြီ။ အောက်ပါလိပ်စာမှာ Download ရယူနိုင်ပါတယ်။

https://nodejs.org

Windows အသုံးပြုသူတွေအနေနဲ့ ပုံမှာပြထားသလို Download စာမျက်နှာကိုဝင်လိုက်ရင် ဦးစားပေးပြထားတဲ့ Container နည်းပညာကို အသုံးပြုတတ်ရင် သုံးပါ။ အသုံးမပြုတတ်ရင် အောက်နားက Installer ဖိုင်ကို ဒေါင်းပြီးတော့သာ Install လုပ်လိုက်ပါ။



မဖြစ်မနေလိုအပ်ချက်မဟုတ်ပေမဲ့ Windows User တွေက Windows Terminal App ကိုလည်း ထည့်ထားသင့်ပါတယ်။

https://apps.microsoft.com/detail/9n0dx20hk701?hl=en-GB&gl=MM

Windows Terminal App မရှိရင် Command Prompt ကိုပဲ အသုံးပြုနိုင်ပါတယ်။ အဆင်ပြေတာကို သုံးပါ။ ဒီနေရာမှာတော့ ရှေ့ဆက်ပြီး Terminal လို့ပဲ သုံးနှုန်းဖော်ပြသွားတော့မှာပါ။

Install လုပ်ပြီးရင် Terminal App မှာ ဒီ Command ကို Run ကြည့်သင့်ပါတယ်။

```
node -v
```

Error တွေဘာတွေမရှိဘူးဆိုရင် အသင့်ဖြစ်နေပါပြီ။ အကယ်၍ node Command ကို မသိဘူးဆိုတဲ့ Error မျိုး ရှိနေရင်တော့ Install လုပ်တာ အဆင်မပြေလို့ပါ။ ပြန် Install လုပ်ဖို့ လိုနိုင်ပါတယ်။

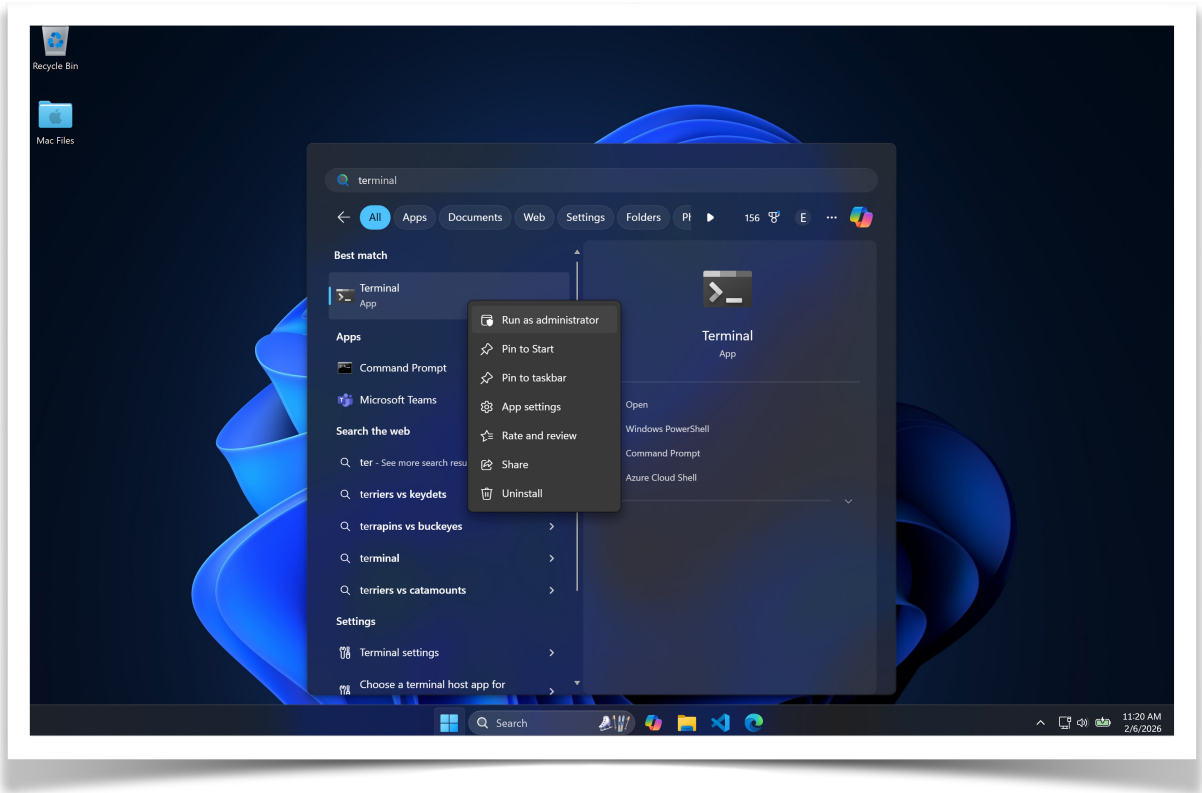
နောက်တစ်ဆင့်အနေနဲ့ ဒီ Command ကို Run ကြည့်ပါ။

```
npm -v
```

Windows မှာ အခုလို Error တက်နေတာမျိုး ဖြစ်တတ်ပါတယ်။

```
npm : File C:\Program Files\nodejs\npm.ps1 cannot be loaded
because running scripts is disabled on this system. For

more information, see about_Execution_Policies at https:/
go.microsoft.com/fwlink/?LinkID=135170.
```

အဲ့ဒီလို Error ဖြစ်နေရင် Terminal App ကို အခုလို Admin အနေနဲ့ဖွင့်ပြီး -



ဒီ Command ကို Run ပေးရမှာ ဖြစ်ပါတယ်။

```
Set-ExecutionPolicy RemoteSigned
```

ဒါက Windows PowerShell မှာ NPM command Run တာကို ခွင့်ပြုအောင် လုပ်ပေးလိုက်တာပါ။ ပြီးတဲ့အခါ အခုလို နောက်တစ်ကြိမ်ပြန်စမ်းကြည့်ပါ။ အဆင်ပြေသွားပါလိမ့်မယ်။

```
npm -v
```

ဒီလောက်ဆိုရင် Setup ပိုင်းပြည့်စုံသွားပါပြီ။

Web Frontend ပရောဂျက်တစ်ခုကို React နဲ့ တည်ဆောက်လိုရင် ပရောဂျက်တည်ဆောက်လိုတဲ့ နေရာမှာ Right Click နှိပ်ပြီး Terminal ကို ဖွင့်လိုက်ပါ။

ပြီးတဲ့အခါ ဒီ Command ကို Run ပေးပါ။

```
npm create vite web-app
```

နည်းပညာရွေးခိုင်းတဲ့အခါ **React** နဲ့ **TypeScript** ကို ရွေးပေးလိုက်ရင် ရပါပြီ။ `web-app` အမည်နဲ့ ပရောဂျက်ဖိုဒါတစ်ခုကို တည်ဆောက်ပေးသွားပါလိမ့်မယ်။

နောက်တစ်ဆင့်အနေနဲ့ လက်ရှိ Terminal ကိုပိတ်လိုက်ပြီး အဲ့ဒီပရောဂျက်ဖိုဒါထဲကိုဝင်ပါ။ Right Click နှိပ်ပြီး Terminal သစ်တစ်ခုကို ဖွင့်ပါ။ ပြီးတဲ့အခါ ဒီ Command တွေကို သုံးပြီး ပရောဂျက်ကို Run ကြည့်လို့ရပါတယ်။

```
npm install
npm run dev
```

`npm install` ဆိုတာ ပရောဂျက်အတွက် လိုအပ်တဲ့ Package တွေကို Download ရယူလိုက်တာပါ။ `npm run dev` ကတော့ ပရောဂျက်ကို Run လိုက်တာဖြစ်ပြီး၊ Run ပြီးတဲ့အခါ သူက စမ်းလို့ရတဲ့ URL လိပ်စာတစ်ခု ထုတ်ပေးလာပါလိမ့်မယ်။ သူထုတ်ပေးတဲ့ လိပ်စာကနေတဆင့် Browser မှာ ဖွင့်ကြည့်ပြီး စမ်းလို့ရပါပြီ။

ရေးထားတဲ့ ကုဒ်တွေကို နောက်ဆုံးဖိုင်နယ် ဖြန့်ဝေလိုရင်တော့ ဒီ Command ကို Run ပေးရပါတယ်။

```
npm run build
```

လက်ရှိပရောဂျက်ဖိုဒါထဲမှာပဲ `dist` အမည်နဲ့ အသင့် Run လို့ရတဲ့ ရလဒ်ကို ထုတ်ပေးသွားပါလိမ့်မယ်။ အများအသုံးပြုဖို့ ဖြန့်ဝေတဲ့အခါ အဲ့ဒီ `dist` ဖိုဒါထဲက ကုဒ်တွေကို ဖြန့်ဝေရတာပါ။ ရေးထားတဲ့ မူလကုဒ်တွေကို ဖြန့်ဝေစရာ မလိုပါဘူး။

ဒီလောက်ဆိုရင်ရပါပြီ။ ပရောဂျက်တွေ ဘယ်လို ဖန်တီးရသလဲ၊ ဘယ်လို Run ရသလဲဆိုတာကို သိသွားတာပါ။ **အခန်း (၆)** မှာ ဒီနည်းတွေကိုသုံးပြီး လက်တွေ့တစ်ခု လုပ်ကြည့်ကြတဲ့အခါ Mobile App ပရောဂျက်တွေ ဘယ်လိုလုပ်ရလဲ ဆိုတာပါ ထည့်ကြည့်ကြပါမယ်။

Platform

နောက်တစ်ဆင့်အနေနဲ့ အဲ့ဒီ Frontend တွေ Backend တွေ ချိတ်ဆက်အလုပ်လုပ်တဲ့ နည်းပညာတွေအကြောင်း ပြောပြချင်ပါတယ်။ အဲ့ဒါတွေကတော့ -

- Client
- Server
- HTTP
- API
- Database နဲ့
- Authentication

- တို့ဖြစ်ပါတယ်။

Client ဆိုတာ Frontend ကိုပဲ ပြောနေတာပါ။ ဒါပေမဲ့ အဓိပ္ပာယ်ကွဲသွားပါတယ်။ Frontend ဆိုရင် အသုံးပြုသူ User နှင့် ထိတွေ့ရသောအပိုင်း လို့ ဆိုလိုပြီး Client ဆိုရင်

တော့ Server ကို ဆက်သွယ်မှုပြုလုပ်ပေးသော နည်းပညာ ဆိုတဲ့ အဓိပ္ပာယ်ဖြစ်သွားပါတယ်။

Server ဆိုတာလည်း အများအားဖြင့် Backend ကိုပဲ ပြောနေတာပါ။ အလားတူ အဓိပ္ပာယ်ကွဲပြားသွားပါတယ်။ Backend ဆိုရင် နောက်ကွယ်က အဓိကအလုပ်လုပ်သော အစိတ်အပိုင်းဆိုတဲ့ အဓိပ္ပာယ်ဖြစ်ပြီး Server ဆိုရင် Client ဆက်သွယ်မှုကို လက်ခံအလုပ်လုပ်ပေးနိုင်သော နည်းပညာ ဆိုတဲ့ အဓိပ္ပာယ် ဖြစ်သွားပါတယ်။

ဒါကြောင့် Mobile App တွေ Web App တွေဟာ Server ကို ဆက်သွယ်မှုပြုမဲ့ Client တွေ ဖြစ်ကြပြီး Database Backend ကတော့ လက်ခံအလုပ်လုပ်ပေးမယ့် Server ဖြစ်သွားပါတယ်။

HTTP ဆိုတာ HyperText Transfer Protocol ရဲ့ အတိုကောက်ဖြစ်ပြီးတော့ Client နဲ့ Server တို့ အသုံးပြုကြတဲ့ ဆက်သွယ်ရေးနည်းပညာပါ။ HTTPS ဆိုရင် HTTP ကိုပဲ သူများတွေ အချက်အလက် ကြားဖြတ်ကြည့်လို့မရတဲ့ လုံခြုံရေးအစီအမံ ပါသွားတာပါ။ ဒါကြောင့် တကယ့်အင်တာနက်ပေါ်မှာ HTTPS ပဲ သုံးကြပါတယ်။

API ဆိုတာ Application Program Interface ရဲ့ အတိုကောက်ဖြစ်ပြီး Backend Server က သတ်မှတ်ထားတဲ့ သူ့ရဲ့ လုပ်ဆောင်ချက်တွေကို ရယူအသုံးပြုနည်း ဖြစ်ပါတယ်။ Frontend Client က API ကို အသုံးပြုပြီး Backend Server ရဲ့ လုပ်ဆောင်ချက်တွေကို ချိတ်ဆက်အသုံးပြုရတာပါ။

Database ဆိုတာ အများအားဖြင့် အချက်အလက်ဒေတာတွေကို ဗဟိုဆာဗာတစ်လုံးနဲ့ သိမ်းဆည်းပေးထားနိုင်တဲ့ နည်းပညာ ပါ။ သူကိုယ်တိုင်က Backend ကဲ့သို့ အလုပ်လုပ်တာ ဖြစ်နိုင်သလို၊ တခြား Backend Server နည်းပညာတခုခုနဲ့ ပူးတွဲအလုပ်လုပ်တာလည်း ဖြစ်နိုင်ပါတယ်။

SQLite လို့ခေါ်တဲ့ ဗဟိုဆာဗာနဲ့မသိမ်းဘဲ ဖိုင်တစ်ခုအနေနဲ့ သိမ်းတဲ့ Database နည်းပညာတွေ ရှိပါသေးတယ်။ မူကွဲ Database ပေါင်းများစွာရှိလို့ ကျွမ်းကျင်သူတွေကတော့ ကိုယ့်ပရောဂျက်နဲ့ လိုက်ဖက်တဲ့ Database အမျိုးအစားကို ရွေးချယ် အသုံးပြုကြပါတယ်။ အခု Vibe Coding လုပ်ကြသူတွေ အသုံးပြုဖို့ တိုက်တွန်းတဲ့ Supabase က PostgreSQL လို့ခေါ်တဲ့ Database နည်းပညာကို အသုံးပြုထားပါတယ်။

Authentication ဆိုတာ သက်ဆိုင်သူကိုပဲ ဝင်ခွင့်ပေးပြီး မသက်ဆိုင်သူကို ဝင်ခွင့်မပေးတဲ့ လုံခြုံရေးအစီအမံပါ။ User Login စနစ်တွေကို ပြောနေတာပါပဲ။ Frontend မှာရော Backend မှာပါ စီစစ်ရပါတယ်။ Frontend မှာပဲ ဟန်ပြစစ်ထားပြီး Backend မှာ မစစ်မိလို့ နောက်ပေါက်က ဟောင်းလောင်းဖြစ်နေတဲ့ Vibe Coding App ပေါင်းများစွာ ရှိကြပါလိမ့်မယ်။ အဲ့ဒီလို မဖြစ်ဖို့လိုပါတယ်။

Authorization ခေါ် ကိုယ်နဲ့ သက်ဆိုင်ရာ အလုပ်ကိုသာ လုပ်ခွင့်ပေးပြီး မသက်ဆိုင်တဲ့ အလုပ်တွေကို လုပ်ခွင့်မပေးတဲ့ အစီအမံ လည်း လိုအပ်ပါသေးတယ်။ ဒီနှစ်ခုက အတွဲလိုက်ပါ။

API ကိုအသုံးပြုတဲ့ စနစ်တွေမှာ ဝင်ခွင့်ရှိမရှိ၊ လုပ်ခွင့်ရှိမရှိ Secret Key တွေနဲ့ ဆုံးဖြတ်ကြပါတယ်။ **Access Token** လို့လည်း ခေါ်ပါတယ်။ Access Token ရှိရင် ဝင်ခွင့်ပေးတယ်၊ မရှိရင် ဝင်ခွင့်မပေးဘူးဆိုတာမျိုးနဲ့ စစ်ကြပါတယ်။

အလားတူပဲ Token ရှိမှ လုပ်ခွင့်ပေးမယ်၊ မရှိရင် လုပ်ခွင့်မပေးဘူးဆိုတဲ့ စနစ်တွေ သုံးကြပါတယ်။ ဘယ်လိုလူကို Token ထုတ်ပေးမှာလဲဆိုတာကတော့၊ အများအားဖြင့် Username / Password မှန်အောင်ပေးသူကို ထုတ်ပေးကြပါတယ်။ ပရောဂျက်ပေါ်မူတည်ပြီး တခြားကန့်သတ်ချက်တွေလည်း ရှိနိုင်ပါတယ်။

ဒီလောက်ဗဟုသုတတွေ ရှိထားရင် ဘာမှမသိဘဲ လက်တမ်းလုပ်တာထက် အများကြီး သာသွားပါပြီ။ အပေါ်ယံသဘောတရားတွေကို ခြုံပြောလိုက်တာဖြစ်လို့ အတွင်းပိုင်းမှာ လေးနက်ကျယ်ပြန့်တာတွေ အများကြီး ကျန်ပါသေးတယ်။ နှစ်ပေါက်အောင် လေ့လာယူရတဲ့ သဘောတရားတွေပါ။ တဖြည်းဖြည်း ပူးတွဲလေ့လာသွားကြရမှာ ဖြစ်ပါတယ်။

ဒီအခန်းက အရေးကြီးပါတယ်။ Coding ပိုင်းမရတဲ့သူတွေအတွက် Code တွေမပါဘဲ သိသင့်တာတွေ စုပြောပြထားတာပါ။ လိုအပ်ရင် နောက်တစ်ကြိမ်လောက် ပြန်ဖတ်ပါလို့ တိုက်တွန်းပါတယ်။

AI နဲ့ စကားပြောတဲ့အစား ဒီအသုံးအနှုန်းတွေ ဗဟုသုတတွေက တော်တော်အသုံးဝင်လာတာကို တွေ့ရပါလိမ့်မယ်။ သူဘာပြောနေတာလဲ ကိုယ်ပိုနားလည်သွားမှာ ဖြစ်သလို၊ ကိုယ်က သိချင်တာတွေကိုလည်း ပိုထဲထဲဝင်ဝင် မေးမြန်းတိုင်ပင်နိုင်သွားမှာပါ။

Coding မရလည်း Vibe Coding လုပ်လို့ရတယ်ဆိုလို့ လုပ်ကြည့်လိုက်တာ ရတစ်ဝက်မရတစ်ဝက်နဲ့ ရုပ်သွားတယ်၊ ဘာဆက်လုပ်ရမှန်းမသိတော့ဘူး ဆိုရင် ဆက်လုပ်ရမှာက ဒါတွေပါပဲ။

အကြောင်းအရာတစ်ခုချင်းစီကို AI နဲ့ အမေးအဖြေလုပ်ပြီး ဆက်လေ့လာသွားရင်တောင် တော်တော်လေး ခရီးရောက်နိုင်ပါတယ်။ စာအုပ်တွေဖတ်ပြီးလေ့လာချင်ရင်တော့ ဒီလင့်မှာ ရယူနိုင်ပါတယ်။ အများစုကို အခမဲ့ပေးထားပါတယ်။

https://eimaung.com

သင်တန်းတက်ပြီး လေ့လာချင်ရင်တော့၊ စာရေးသူကိုယ်တိုင် ပူးတွဲတည်ထောင်ထားပြီး (၁၅) နှစ်လောက် သက်တမ်းရှိပြီဖြစ်တဲ့ **Fairway Technology** ခေါ်သင်တန်းကျောင်း ရှိပါတယ်။ ဒီလင့်မှာ လေ့လာကြည့်နိုင်ပါတယ်။

http://fairwayweb.com

အခန်း (၆) - အပျော်တမ်းထက်ပိုသောပရောဂျက်

ဒီအခန်းမှာ စမ်းသပ်ယုံသက်သက် မဟုတ်တော့ဘဲ လက်တွေ့အသုံးချဖို့ထိ ရည်ရွယ်ထားတဲ့ ပရောဂျက်တွေကို လုပ်တဲ့အခါ ဘယ်လိုအဆင့်တွေနဲ့ ဘယ်လိုလုပ်သွားလေ့ ရှိသလဲ ဆိုတာကို လုပ်ပြမှာ ဖြစ်ပါတယ်။

Book Club App လေးတစ်ခု လုပ်ချင်တာပါ။ သူငယ်ချင်းတွေ စာအတူတူဖတ်ဖို့ပါ။

Coding Agent အနေနဲ့ **Cursor** ကို အသုံးပြုဖို့ ရွေးချယ်ထားပါတယ်။ စာဖတ်သူက လိုက်ပြီး စမ်းကြည့်ချင်တယ်ဆိုရင် Cursor, Antigravity, Codex, Claude Code စသည်ဖြင့် မည်သည့် နည်းပညာနဲ့မဆို စမ်းကြည့်လို့ရပါတယ်။

ဘယ်ဟာပိုကောင်းတဲ့၊ ဘာနဲ့ပဲ လုပ်သင့်တယ်ဆိုတဲ့ အစွဲအလမ်းတွေ ထားစရာမလိုပါဘူး။ လက်တွေ့မှာ အားလုံးကို သုံးကြပါတယ်။ အသုံးပြုနည်းတွေက တစ်ခုနဲ့တစ်ခု ခပ်ဆင်ဆင်ပါပဲ။ နှစ်သက်ရာကို အသုံးပြုနိုင်ပါတယ်။

နောက်ဆုံးဆင့် ဖုန်းနဲ့ပဲ Google AI Studio သုံးပြီး လိုက်လုပ်မယ်ဆိုရင်တောင် Web App ပိုင်းထိ လုပ်လို့ရပါတယ်။ လိုအပ်ရင် Cursor ကို အောက်ပါလိပ်စာမှာ Download ရယူနိုင်ပါတယ်။

https://cursor.com

Planning

ပထမဆုံး ဘာလိုချင်တာလဲ၊ ဘာတွေသုံးပြီး ဘယ်လိုချိတ်ဆက် အလုပ်လုပ်သင့်လဲဆိုတာ ကိုယ့်ခေါင်းထဲမှာ ရှင်းရပါမယ်။ AI ကို “**ဒါလေးလုပ်ပေးပါ**” နဲ့ စတယ်ဆိုတာ အပျော်တမ်းအဆင့်ပဲ ရပါမယ်။ တကယ်လက်တွေ့ အသုံးချအဆင့် ရောက်ချင်ရင် အများကြီး ကြိုတင်ပြင်ဆင်ရပါမယ်။

ပထမဆုံးအနေနဲ့ နည်းပညာပိုင်း စဉ်းစားရပါမယ်။ ဒီပရောဂျက်မှာ Web App ရော Mobile App ပါ ထည့်လုပ်ဖို့ ဆုံးဖြတ်ပါတယ်။ Mobile App ဆိုတဲ့နေရာမှာလည်း Android ရော iOS ပါ နှစ်ခုလုံးကို Support လုပ်ကြပါမယ်။ Backend အနေနဲ့ကတော့ Supabase ကိုပဲ သုံးလိုက်ကြပါမယ်။

ကိုယ်တိုင်သာ ဆိုရင်တော့ Language, Library, Framework, Components, Routing, Form စသည်ဖြင့် အသေးစိတ် ကြိုတင်ရွေးချယ်မှာပါ။ ဒီနေရာမှာတော့ ဒီလိုအသေးစိတ်တွေ မသိရင်လည်း လုပ်လို့ရနိုင်တာကို ပြချင်တဲ့အတွက် အဲ့ဒီလောက်ထိ မသွားတော့ပါဘူး။

နောက်တစ်ဆင့်အနေနဲ့ လိုချင်တဲ့အလုပ်လုပ်ပုံ စဉ်းစားရပါမယ်။

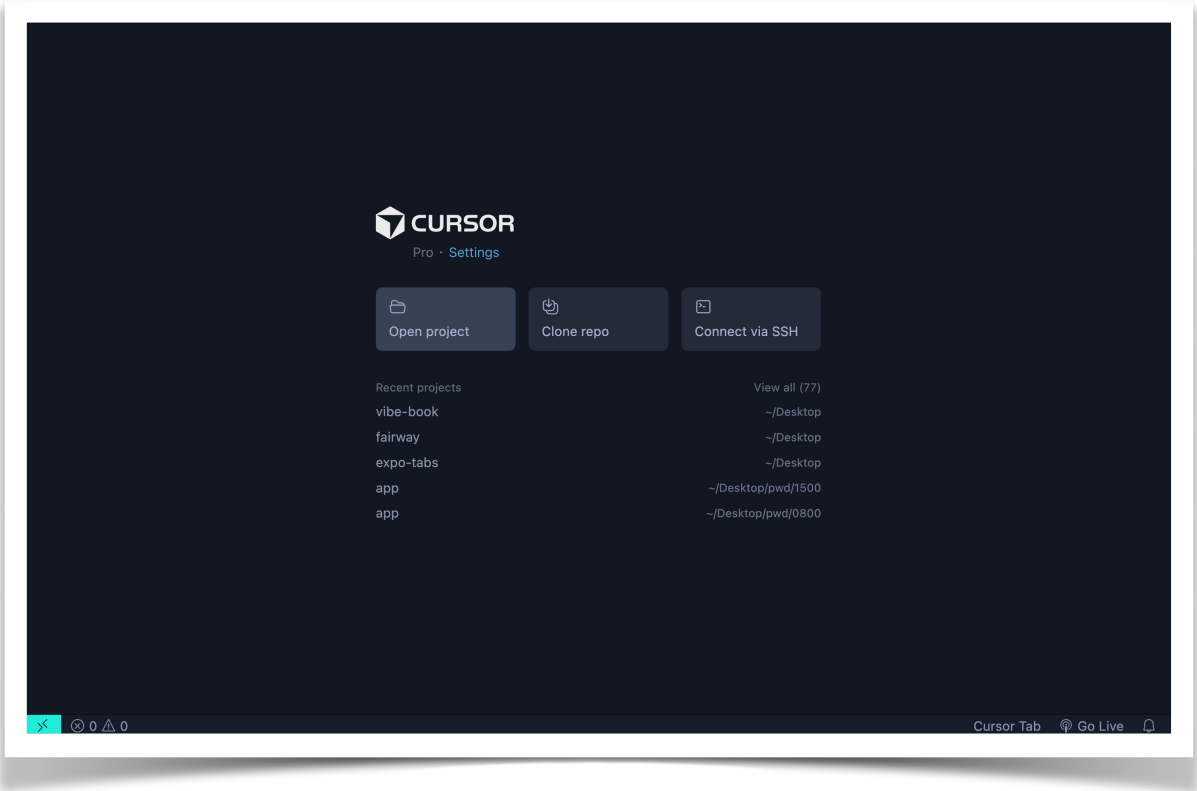
- User တွေက Reading Session သစ်ဖွင့်လို့ရရမယ်။
- စာအုပ်အမည်၊ ရေးသားသူနဲ့၊ ပါဝင်တဲ့အခန်းတွေ တင်လို့ရမယ်။
- တခြားသူတွေက အဲ့ဒီ Session ကို Join လို့ရမယ်။
- ဘယ်အခန်းထိ ဖတ်ပြီးပြီလဲဆိုတာ Submit လုပ်လို့ရမယ်။
- ဒါကြောင့် ဘယ်သူက ဘယ်အခန်းထိ ဖတ်လို့ ပြီးသွားပြီလဲဆိုတဲ့ Progress တွေလည်း ကြည့်လို့ရရမယ်။
- ဖတ်နေစဉ်နဲ့ ဖတ်ပြီးနောက် သူ့အမြင်ကိုယ့်အမြင် တိုင်ပင်လို့ရမယ်။
- Reaction လုပ်ဆောင်ချက် ပါချင်တယ်။

ဒီလိုမျိုး ကိုယ်ဖြစ်ချင်တာ ကိုယ်လိုချင်တာတွေ အားလုံးကို အသေးစိတ် ရေးသားမှတ်တမ်းတင်ထားတဲ့ **Project Spec** နဲ့ စသင့်ပါတယ်။ တော်တော်အရေးကြီးပြီး အသုံးဝင်တယ်ဆိုတာကို နောက်ပိုင်းမှာ တွေ့လာပါလိမ့်မယ်။

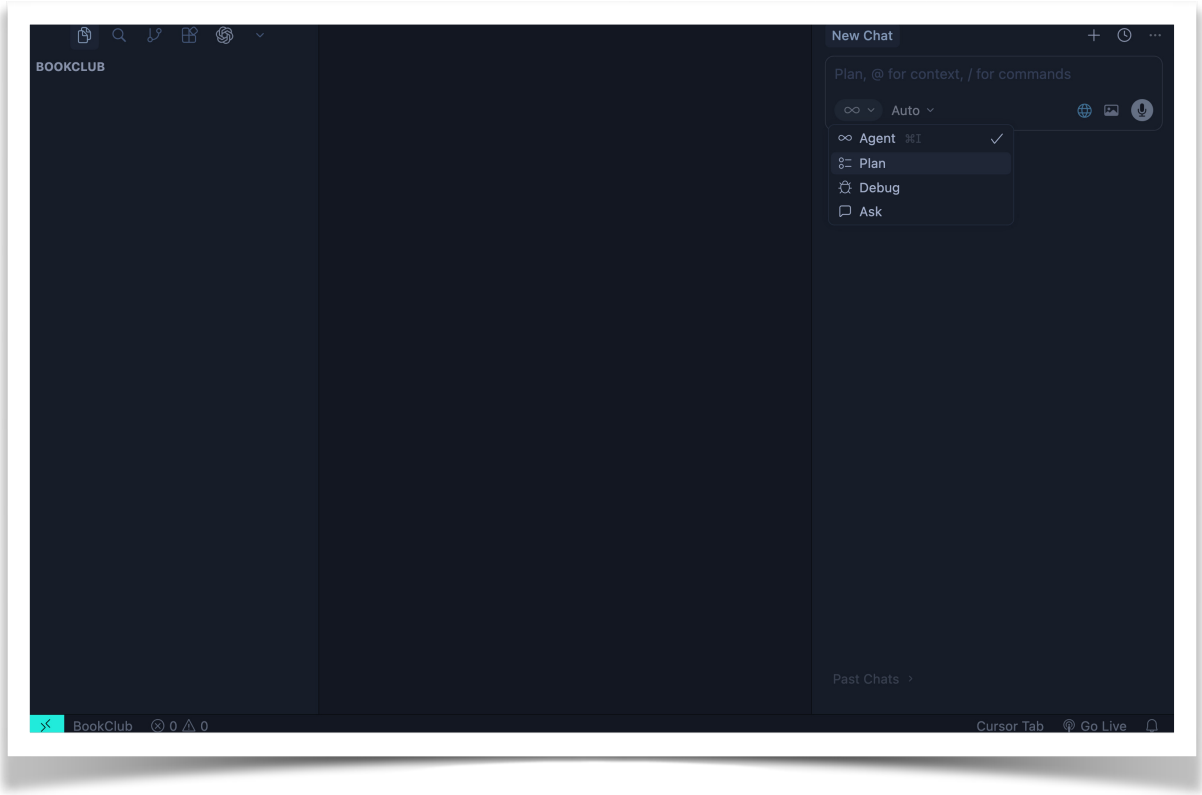
Project Spec

Cursor လို Agent မျိုးမှာ **Plan Mode** ပါပါတယ်။ ဒီလို ကိစ္စမျိုးအတွက် အထူး အသုံးဝင်ပါတယ်။ ဒါပေမဲ့ တချို့ Agent တွေမှာ Plan Mode မရှိလို့ နမူနာမှာ Plan Mode ကို သုံးမပြတော့ဘဲ ရိုးရိုး Agent နဲ့ပဲ လုပ်ပြပါမယ်။

ပထမဦးဆုံး ကိုယ့်ကွန်ပျူတာထဲက နှစ်သက်ရာနေရာမှာ ဖိုဒါတစ်ခု ဆောက်လိုက်ပါ။ ပြီးတဲ့အခါ Cursor ရဲ့ **Open Project** ခလုတ်ကိုနှိပ်ပြီး အဲ့ဒီဖိုဒါကို ဖွင့်လိုက်ပါ။



စစချင်းတွေ့မြင်ရတာက ဘယ်ဘက်ခြမ်းမှာ Primary Sidebar ဖြစ်ပြီး ပရောဂျက်မှာပါတဲ့ ကုဒ်ဖိုင်တွေကို ပြပေးပါလိမ့်မယ်။ ညာဘက်ခြမ်းမှာ Secondary Sidebar ဖြစ်ပြီး Agent Chat ကိုပြပေးပါလိမ့်မယ်။

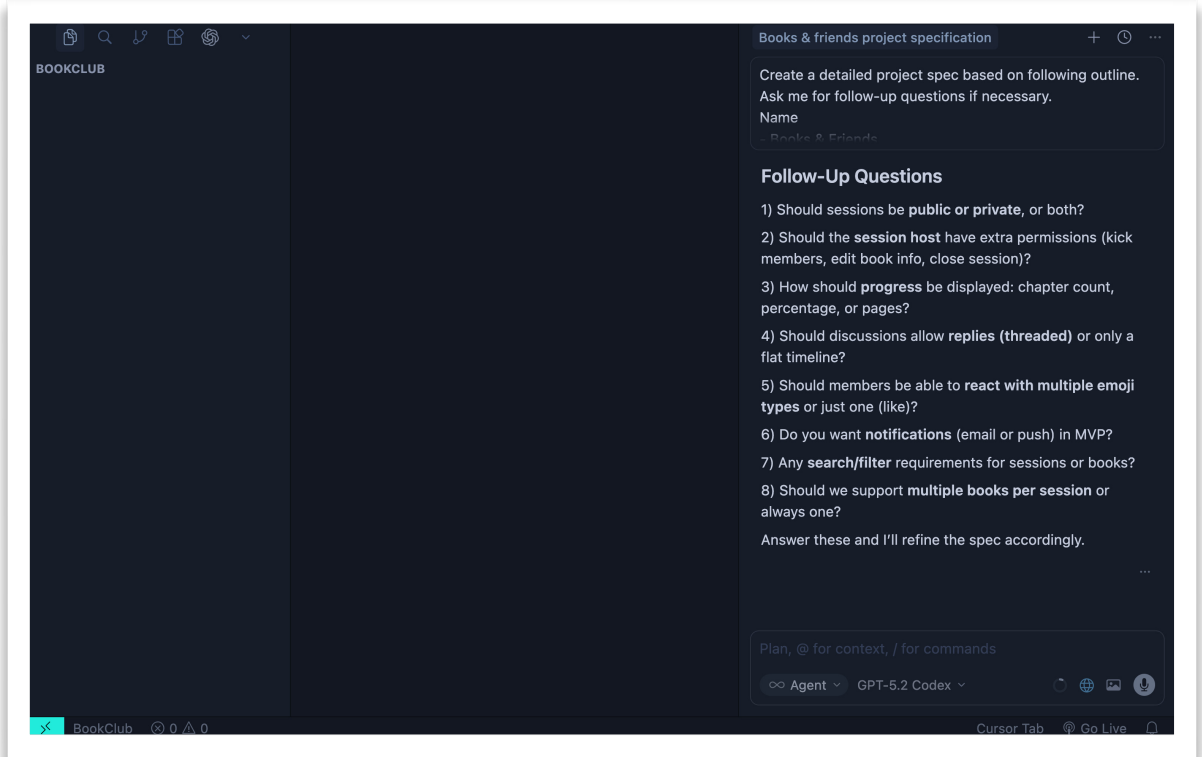


စောစောကပြောခဲ့သလိုပဲ Plan Mode ရှိပေမဲ့ မသုံးတော့ဘဲ ရိုးရိုး Agent နဲ့ဘဲ ဆက်သွားလိုက်ပါမယ်။ Model ကတော့ Codex ကို သုံးထားပါတယ်။ အကယ်၍ Pro Plan မဟုတ်လို့ Codex Model မရရင်လည်း Auto ကို အသုံးပြုနိုင်ပါတယ်။ ရတဲ့ Model တွေထဲက လုပ်ခိုင်းတဲ့အလုပ်နဲ့ သင့်တော်ရာကို အလိုအလျောက် ရွေးချယ် အသုံးပြုပေးသွားပါလိမ့်မယ်။

ပထမဆုံးအနေနဲ့ စောစောကပြောခဲ့တဲ့ အချက်အလက်တွေ ပါဝင်တဲ့ ကိုယ့်ပရောဂျက်ရဲ့ Spec အကြမ်းလေး သူ့ကို ပေးလိုက်ပြီး Detail ရေးခိုင်းလိုက်ပါတယ်။

> *Create a detailed project spec based on following outline.* ***Ask me for follow-up questions if necessary****.*
>
> ***Name***
>
> *- Books & Friends*
>
> ***Description***
>
> - *Anyone can register. A member can create reading session for a book by submitting title, author and chapters. Other can join, submit their progress. Member can discuss about it in single thread. Member can react to other members' comments and discussion.*
>
> ***Services***
>
> *- Supabase Backend*
>
> ***Product***
>
> - *Web App*
> - *Mobile App*

Prompt ထဲမှာ "**လိုအပ်တဲ့ မေးခွန်းတွေ ငါ့ကို ပြန်မေးပါ**" လို့ ထည့်ပြောထားတဲ့အတွက် Agent က တော်တော်စိတ်ဝင်စားဖို့ကောင်းပြီး အသုံးဝင်တဲ့ နောက်ဆက်တွဲ မေးခွန်းတွေကို မေးလာပါတယ်။



(စာဖတ်သူလိုက်လုပ်ရင် အတိအကျတော့ တူချင်မှ တူပါလိမ့်မယ်။ ဒါကြောင့် စာရေးသူ လုပ်ခဲ့တာကိုပဲ ဆက်ပြောပြပါမယ်။)

သူက နမူနာပုံမှာ ပြထားသလို မေးခွန်းတွေ မေးလာတာပါ။

မေး - Reading Session တွေဟာ Private ဖြစ်သင့်လား၊ Public ဖြစ်သင့်လား။

ဖြေ - Session ပိုင်ရှင်က ဆုံးဖြတ်ခွင့်ရှိသင့်တယ်လို့ ယူဆပါတယ်။

မေး - Session ပိုင်ရှင်က Member တွေ ပြန်ထုတ်ခွင့်ရှိသင့်လား။

ဖြေ - ရှိသင့်တယ်၊ ဒါပေမဲ့ ပထမ Version မှာ အဲ့ဒါတွေ မထည့်ချင်သေးပါဘူး။

မေး - Progress ကို ဘယ်လိုပြသင့်လဲ။

ဖြေ - Progress Bar လေးနဲ့ ပြသင့်ပါတယ်။ စာအုပ်မှာ ပါဝင်တဲ့ အခန်းအရေအတွက်ကိုပဲ ကြည့်သင့်ပါတယ်။

မေး - Comment ကို ထပ်ဆင့် Reply လုပ်လို့ရသင့်လား။

ဖြေ - ရရင်ကောင်းပေမဲ့ ပထမ Version မှာ မထည့်ချင်သေးပါဘူး။

မေး - Reaction အတွက် Like တစ်ခုပဲ ထည့်မှာလား၊ အစုံထည့်မှာလား

ဖြေ - အစုံထည့်သင့်ပါတယ်။

မေး - Notification တွေဘာတွေ ပါဦးမလား

ဖြေ - ပထမ Version မှာ မထည့်သေးပါဘူး။

မေး - Search တွေဘာတွေ ပါဦးမလား

ဖြေ - ပထမ Version မှာ မထည့်သေးပါဘူး။

မေး - Session တစ်ခုမှာ စာအုပ်တစ်အုပ်ထက် ပိုရင်ရောရလား

ဖြေ - ပထမ Version မှာ Session တစ်ခု စာအုပ်တစ်အုပ်ပဲ ထားပါဦးမယ်။

ဒီနေရာမှာ Agent အကြံပေးတဲ့ Feature တော်တော်များများက ကောင်းပေမဲ့ ငြင်းထားတာကို သတိပြုပါ။ ပထမတစ်ဆင့်မှာ **MVP** ခေါ် Minimal Viable Product နဲ့ စရတာ

ထုံးစံ ဖြစ်ပါတယ်။ အနည်းဆုံး မဖြစ်မနေ လိုအပ်တာတွေနဲ့ အရင်စရပါတယ်။ အကုန်တစ်ခါတည်း မလုပ်ရပါဘူး။ ပရောဂျက်သေးလေးတွေကို တစ်ချက်တည်း အစအဆုံး လုပ်တာ ပြဿနာမရှိပေမဲ့ ပရောဂျက်ကြီးလာရင် မရပါဘူး။ မလုပ်သင့်ပါဘူး။

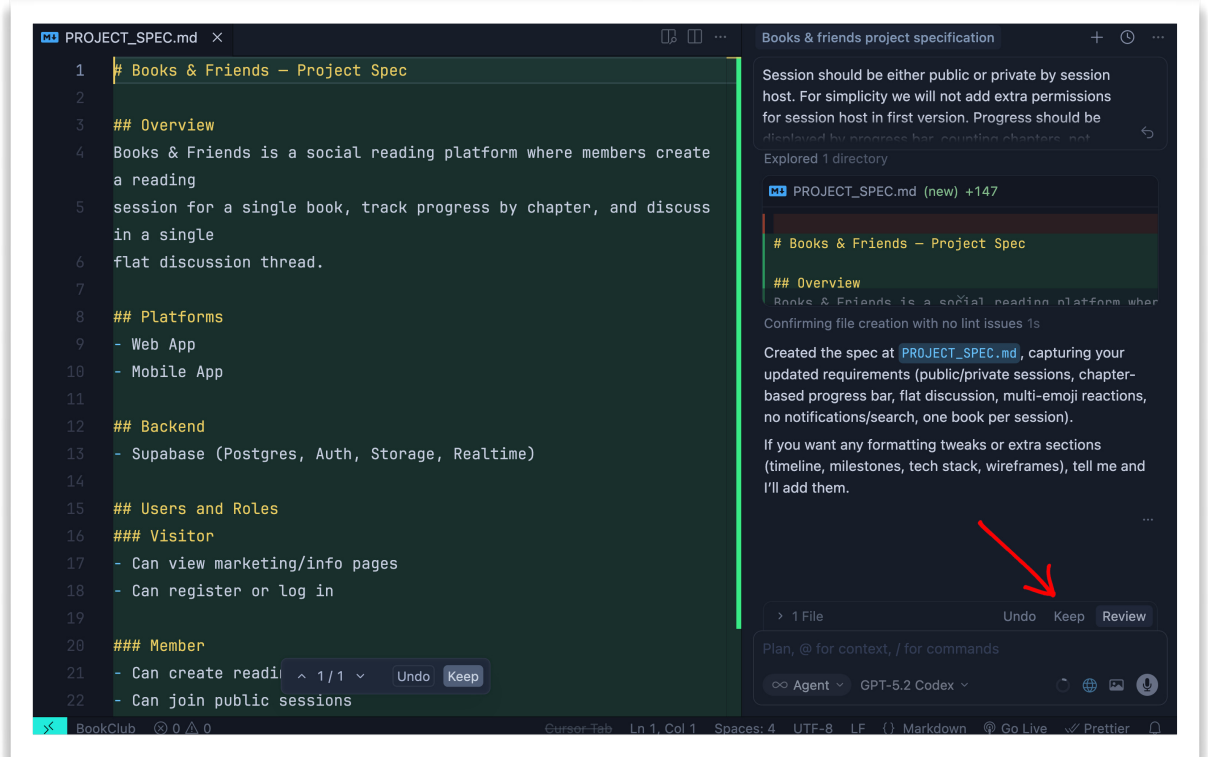
တစ်ဆင့်ချင်းသာ သွားသင့်ပါတယ်။ ဒါကြောင့် အနည်းဆုံး မဖြစ်မနေ ပါစေချင်တာပဲ Spec ထဲမှာ ထည့်ထားပါတယ်။ နောက် Version ကျမှ ထပ်ထည့်မှာတွေလည်း Spec ထဲမှာ ကြိုထည့်ထားလို့ရပေမဲ့ ရည်ရွယ်ချက်ကို ရှင်းအောင် ပြောထားရမှာ ဖြစ်ပါတယ်။

ဒါကြောင့် အခုလို Prompt လေး ထပ်ပေးလိုက်ပါတယ်။

> *Session should be either public or private by session host. For simplicity we will not add extra permissions for session host in first version. Progress should be displayed by progress bar, counting chapters, not pages. For simplicity discussion should be flat. Member should be able to react with multiple emoji. No notification for first version. No search feature for first version. Each session should have only one book.* ***Write spec out to a markdown file****.*

စောစောကသူမေးတဲ့ မေးခွန်းတွေကို ပြန်ဖြေပြီး Detail Spec ကို Markdown ဖိုင်လေး တစ်ခုနဲ့ ရေးမှတ်ခိုင်းလိုက်တာပါ။

ကိုယ်ပြောတဲ့အတိုင်း အားလုံးအပြည့်အစုံပါတဲ့ Project Spec ဖိုင်လေး ရေးပေးသွားပါတယ်။ တချို့ Codeing Agent တွေမှာ ရေးလိုက်တဲ့ ကုဒ်ဖိုင်တွေကို အလိုအလျောက် အတည်ပြုပါတယ်။ Cursor မှာ ကိုယ်က အတည်ပြုပေးရပါတယ်။ ဒါကြောင့် ပုံမှာ မျှားပြထားတဲ့ Keep ခလုတ်လေးကို နှိပ်ပြီး အတည်ပြုပေးရပါတယ်။

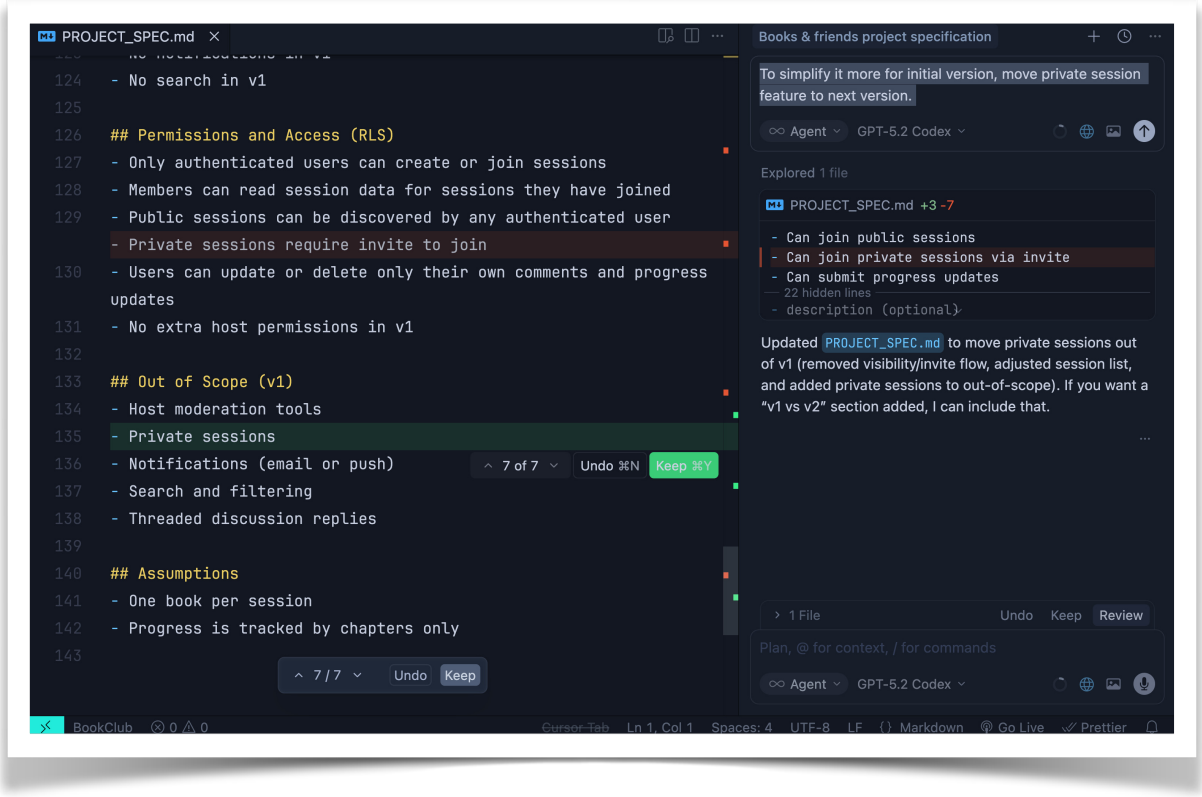


သူရေးပေးတာကို မလိုချင်ရင် Undo လုပ်လို့ရအောင် စီစဉ်ပေးထားတာပါ။ တချို့က ဒီသဘောကို ကြိုက်ကြပါတယ်။ တချို့ကတော့ မကြိုက်ကြပါဘူး၊ အလိုအလျောက် တခါတည်း အကုန်လုပ်ပေးသွားတာပဲ ကြိုက်ကြပါတယ်။

ရေးပေးတဲ့ Spec ကို ဖတ်ကြည့်တော့ Public/Private Session ကိစ္စမှာ နည်းနည်းရှုပ်နေပါတယ်။ အစပိုင်း အဆင်မပြေမှာစိုးလို့ အဲ့ဒီ Feature မထည့်တော့ဘူးလို့ ဆုံးဖြတ်ပြီး အခုလို Prompt ပေးလိုက်ပါတယ်။

> *To simplify it more for initial version,* ***move private session feature to next version****.*

Cursor ရဲ့ နောက်ထပ်ထူးခြားချက်က Agent ရေးပေးတဲ့ ကုဒ်တစ်ပိုင်းချင်းစီကို လူက Review လုပ်လို့ရတာပါပဲ။ စာရေးသူလို ရေးထားတဲ့ကုဒ်တွေ အသေးစိတ် ဖတ်ချင်သူနဲ့ ကိုက်ပါတယ်။ စာဖတ်သူက မဖတ်ချင်ရင် ညာဘက် Agent Chat နားက **Keep** ခလုတ် နှိပ်ပြီးတော့ပဲ လက်ခံပေးလိုက်ရင် ရပါတယ်။

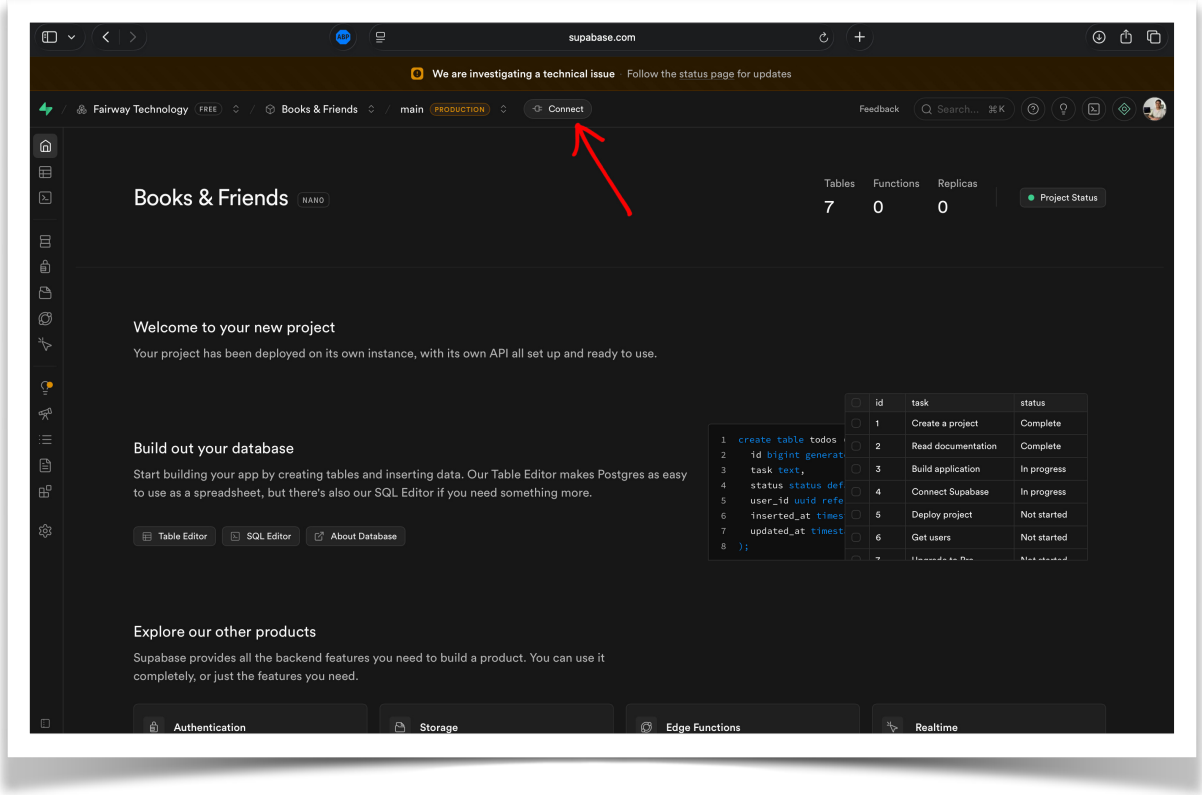


အခုဆိုရင် လက်ရှိ Version မှာ ဘာတွေလုပ်မယ့် နောက် Version ကျမှ ဘာတွေ ထပ်ထည့်မယ်ဆိုတာ အပြည့်အစုံပါတဲ့ Project Spec တစ်ခု ရသွားပါပြီ။

နောက်တစ်ဆင့်မှာ Agent ကို ကုဒ်တွေ မရေးခိုင်းခင် Supabase မှာ လိုအပ်တာ အရင် လုပ်သင့်ပါတယ်။ ဒါကြောင့် Supabase ပရောဂျက်အရင် Create လုပ်သင့်ပါတယ်။

https://supabase.com

ပရောဂျက်အမည်ကို **Books & Friends** လို့ပေးလိုက်ပြီး၊ Connect ခလုတ်ကိုနှိပ်ပါတယ်။ ပြီးခဲ့တဲ့အခန်းမှာ နမူနာပြခဲ့သလို URL နဲ့ API Key တွေ ရယူလိုက်ပါတယ်။



အချက်အလက်တွေစုံပြီမို့လို အခုလို Prompt ပေးလိုက်ပါတယ်။

> *Following is Supabase url [supabase_url] and following is supabase publishable api key [publishable_key]*
>
> *Start implementation. **Only work** on web app first using React with Vite. UI should be simple but lovely. Should work well on various*

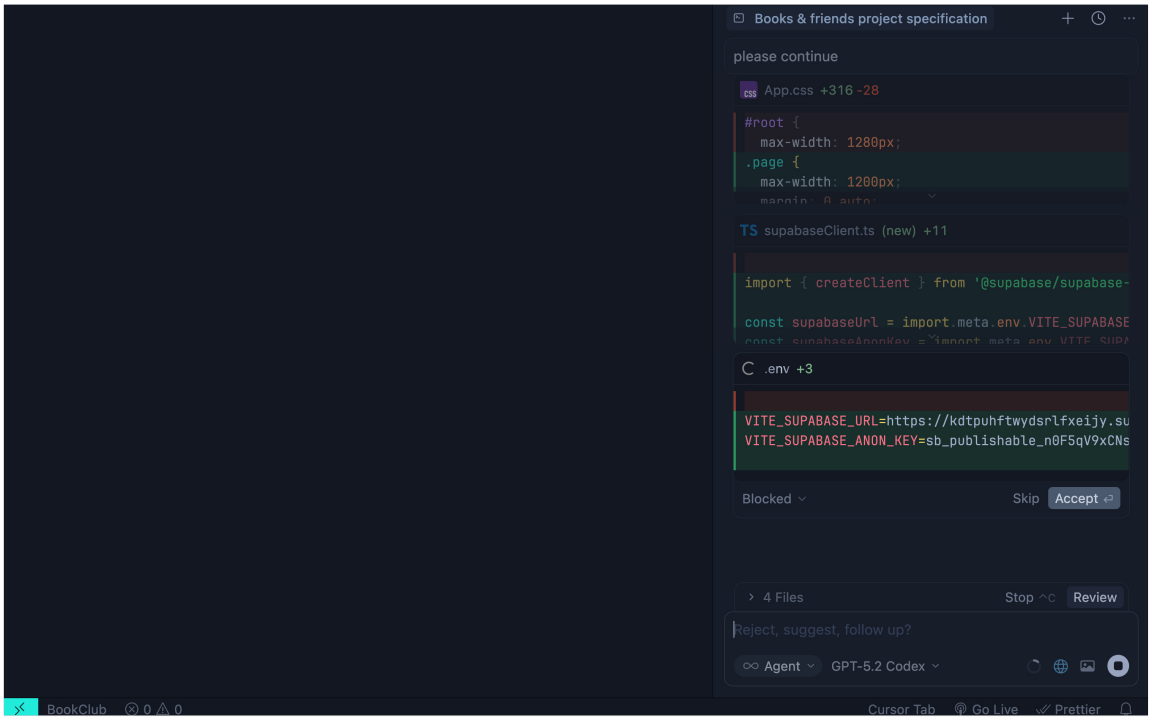
> *screen sizes.* ***Check updated documentation for relevant techs if necessary****.*

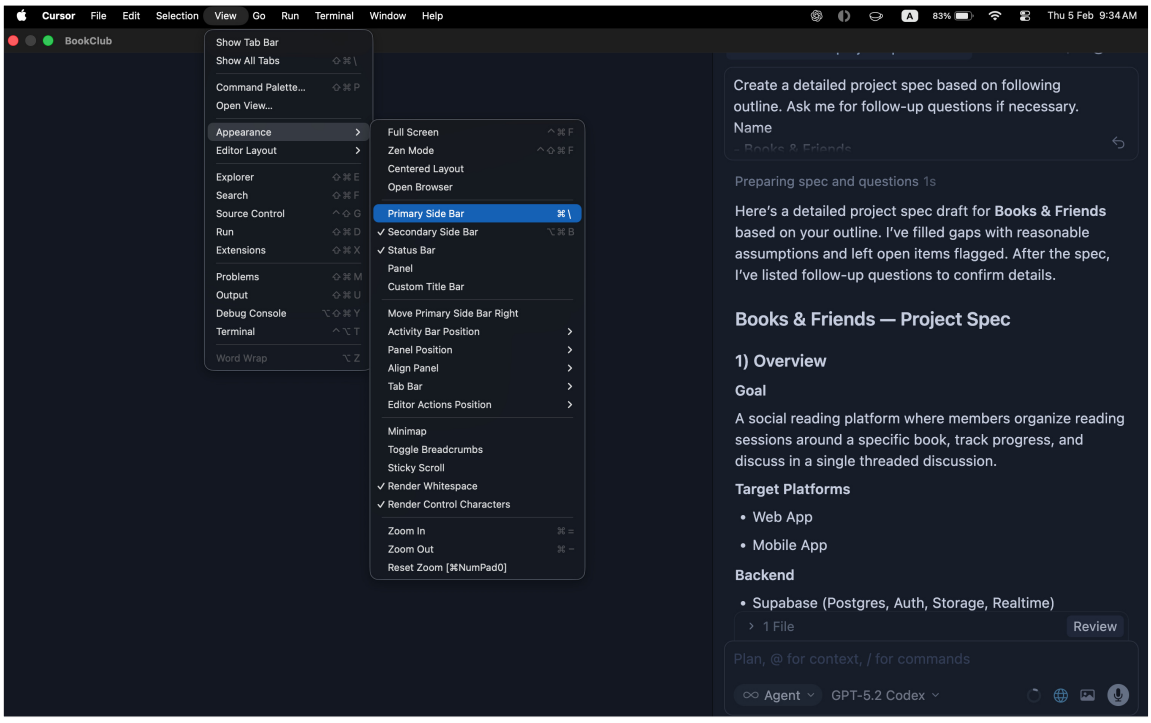
Web App ကိုပဲ အရင်လုပ်ခိုင်းပါတယ်။ Mobile App မလုပ်ခိုင်းသေးပါဘူး။ UI ပိုင်းက ရိုးရိုးလေးနဲ့ ကြည့်ကောင်းတာလေး လိုချင်တယ်လို့ပဲ ပြောလိုက်ပါတယ်။ အရမ်း အသေးစိတ်ပြောမထားပါဘူး။ ပြောချင်ရင်လည်း ရပါတယ်။ ဘယ်လိုအရောင် ကာလာ ဒီဇိုင်းတွေ သုံးစေချင်သလဲ ဆန္ဒရှိရင် ထည့်ပြောလို့ရပါတယ်။ ထုံးစံအတိုင်း `[supabase_url]` နဲ့ `[publishable_key]` နေရာမှာ အချက်အလက်အမှန်ကို အစားထိုး ထည့်ပေးဖို့ လိုပါတယ်။

လိုအပ်ရင် Update Documentation ဖတ်ပါ လို့လည်း ထည့်ပြောထားပါတယ်။ ဒီတော့မှ AI က နည်းဟောင်းတွေမသုံးဘဲ နောက်ဆုံး Update နည်းတွေကို သုံးပြီး အလုပ်လုပ်ပေးမှာပါ။

ဒီ Prompt နဲ့တင် ပထမတစ်ဆင့် ရေးလို့ပြီးသလောက် ဖြစ်နေပါပြီ၊ ဒါပေမဲ့ တချို့ Sensitive ဖြစ်တဲ့ဖိုင်တွေဆိုရင် Cursor က Auto ဖန်တီးမပေးပါဘူး။ ကိုယ့်က Accept လုပ်ပေးရပါတယ်။ ပုံမှာတွေ့မြင်နေရတာက `.env` လို့ခေါ်တဲ့ Setting တွေ Secret Key တွေ သိမ်းလေ့ရှိတဲ့ဖိုင်ကို ရေးပေးထားပြီး ကိုယ့်ဘက်က Accept လုပ်ပေးဖို့ လိုနေတာပါ။ ဒါကြောင့် **Accept** ခလုတ်လေး နှိပ်ပေးလိုက်ပါတယ်။

ဘယ်ဘက်ခြမ်းက Primary Sidebar ကိုလည်း ခဏဖျောက်ထားတာ တွေ့ရနိုင်ပါတယ်။ Screen မှာ ကြည့်ရတာရှုပ်ရင် တချို့ဟာတွေ ခဏဖျောက်ထားပြီး လိုအပ်မှ ပြန်ခေါ်လို့ရပါတယ်။

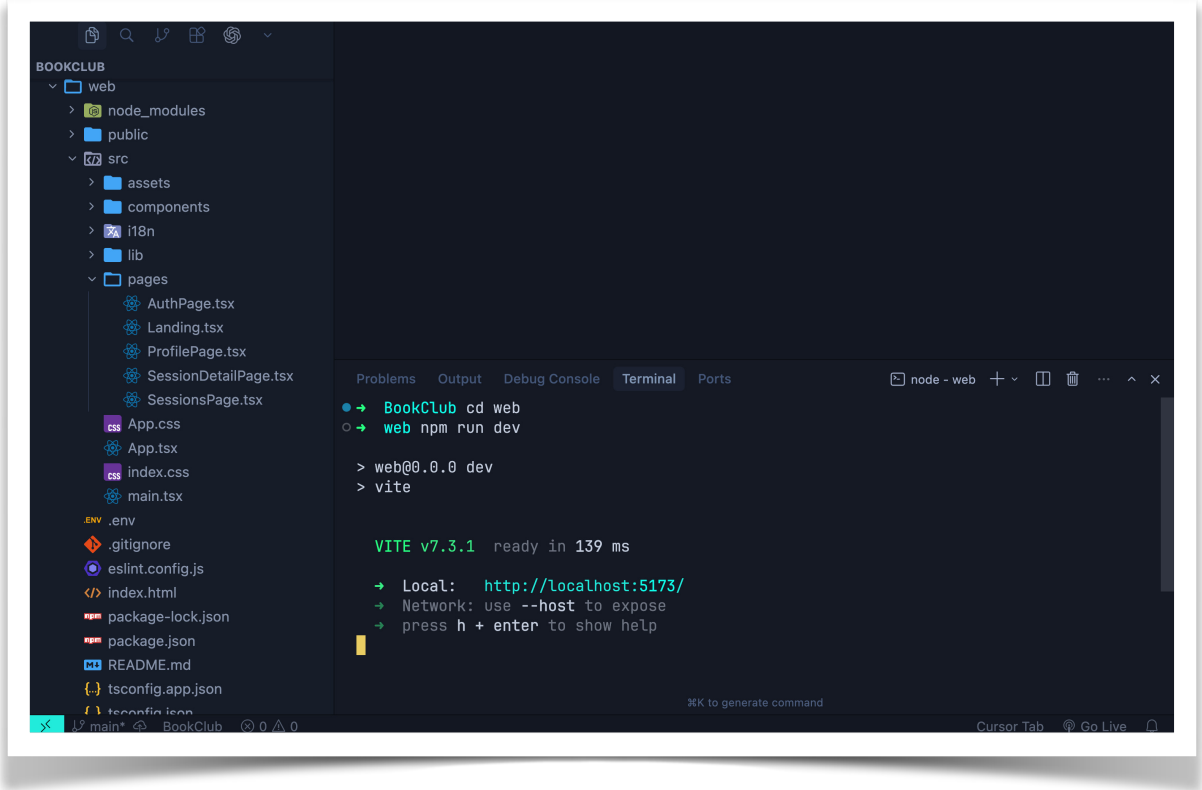

Books & friends project specification
please continue
App.css +316 -28
#root {
max-width: 1280px;
.page {
max-width: 1200px;
TS supabaseClient.ts (new) +11
import { createClient } from '@supabase/supabase-
const supabaseUrl = import.meta.env.VITE_SUPABASE
.env +3
VITE_SUPABASE_URL=https://kdtpuhftwydsrlfxeijy.su
VITE_SUPABASE_ANON_KEY=sb_publishable_n0F5qV9xCNs
Blocked
Skip
Accept
4 Files
Stop ^C
Review
Reject, suggest, follow up?
Agent
GPT-5.2 Codex
BookClub
Cursor Tab
Go Live
Prettier


Cursor File Edit Selection View Go Run Terminal Window Help
Thu 5 Feb 9:34 AM
BookClub
Show Tab Bar
Show All Tabs
Command Palette...
Open View...
Appearance
Editor Layout
Explorer
Search
Source Control
Run
Extensions
Problems
Output
Debug Console
Terminal
Word Wrap
Full Screen
Zen Mode
Centered Layout
Open Browser
Primary Side Bar
Secondary Side Bar
Status Bar
Panel
Custom Title Bar
Move Primary Side Bar Right
Activity Bar Position
Panel Position
Align Panel
Tab Bar
Editor Actions Position
Minimap
Toggle Breadcrumbs
Sticky Scroll
Render Whitespace
Render Control Characters
Zoom In
Zoom Out
Reset Zoom [⌘NumPad0]
Create a detailed project spec based on following outline. Ask me for follow-up questions if necessary.
Name
Preparing spec and questions 1s
Here's a detailed project spec draft for Books & Friends based on your outline. I've filled gaps with reasonable assumptions and left open items flagged. After the spec, I've listed follow-up questions to confirm details.
Books & Friends — Project Spec
1) Overview
Goal
A social reading platform where members organize reading sessions around a specific book, track progress, and discuss in a single threaded discussion.
Target Platforms
Web App
Mobile App
Backend
Supabase (Postgres, Auth, Storage, Realtime)
1 File
Review
Plan, @ for context, / for commands
Agent
GPT-5.2 Codex

View Menu ထဲက **Appearance** ကို သွားပြီး Primary Side Bar တို့ Secondary Side Bar တို့ကို အဖွင့်အပိတ်လုပ်လို့ရပါတယ်။

Agent ကို Prompt တွေပေးရင်းနဲ့ လိုင်းမကောင်းလို့ဘဲဖြစ်ဖြစ် တစ်ဝက်တစ်ပျက် ရပ်သွားတာတွေဘာတွေ ရှိရင် ရောက်တဲ့နေရာမှာ ရပ်လိုက်ပြီး “**ဆက်လုပ်ပါ**” လို့ Prompt ပေးလိုက်လို့ရပါတယ်။ သိပ်ပြဿနာမရှိဘဲ ဆက်လုပ်နိုင်လေ့ရှိပါတယ်။

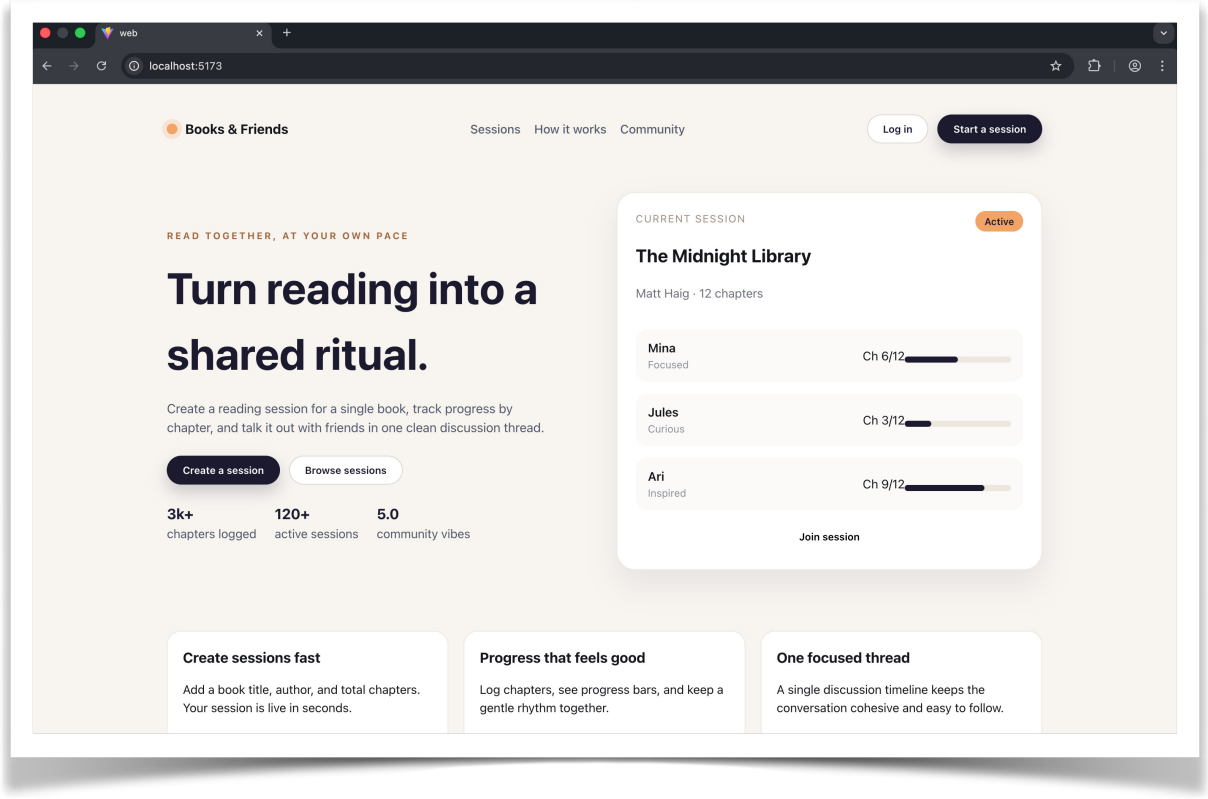
အခုဆိုရင် ပထမတစ်ဆင့် ရေးလို့ပြီးသွားပြီမို့လို့ Run ကြည့်လို့ရပါပြီ။ Run ပုံ Run နည်းကို Agent ကလည်း ပြောပြပါလိမ့်မယ်။ ဆန္ဒရှိရင် သူ့ကိုပဲ Run ခိုင်းလိုက်လို့လဲ ရပါတယ်။



နမူနာမှာတော့ **View** Menu ထဲက **Terminal** ကို နှိပ်ဖွင့်ပြီး အခုလို Command နှစ်ခု Run လိုက်တာပါ။

```
cd web
npm run dev
```

`cd web` ဆိုတာ Web App ပရောဂျက်ထဲကို သွားလိုက်တာပါ။ ပြီးတော့မှ `npm run dev` နဲ့ Run လိုက်ပါတယ်။ ရလာတဲ့ URL ကို နှိပ်ပြီးဖွင့်လိုက်တဲ့အခါ အခုလိုရလဒ်ကို ရရှိပါတယ်။



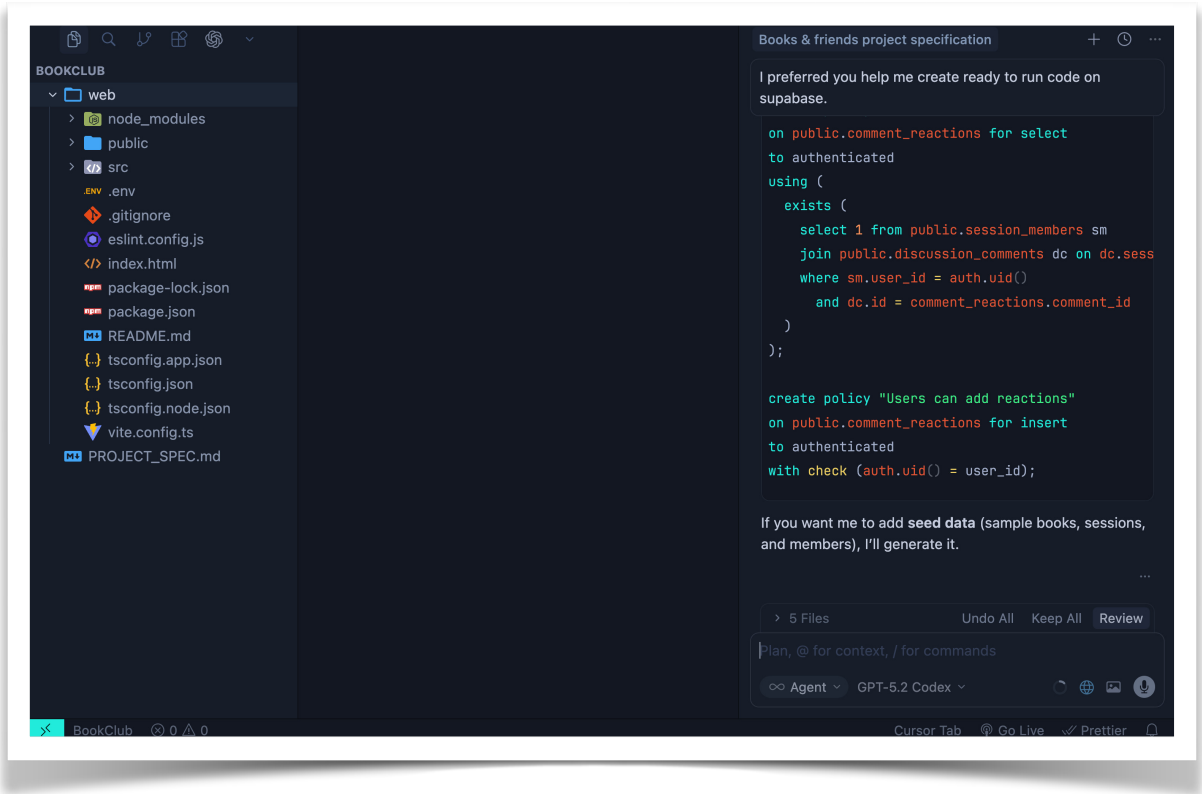
တော်တော်ကို သပ်ရပ်လှပ ကောင်းမွန်တဲ့ ရလဒ်တစ်ခုကို ရလိုက်တာပါ။

ပထမတစ်ဆင့်ပဲ ရှိပါသေးတယ်။ အားလုံး အလုပ်မလုပ်သေးပါဘူး။ နောက်အဆင့်တွေ ဆက်သွားဖို့အတွက် Supabase ဘက်မှာ လုပ်သင့်တာတွေ လုပ်ကြရပါဦးမယ်။

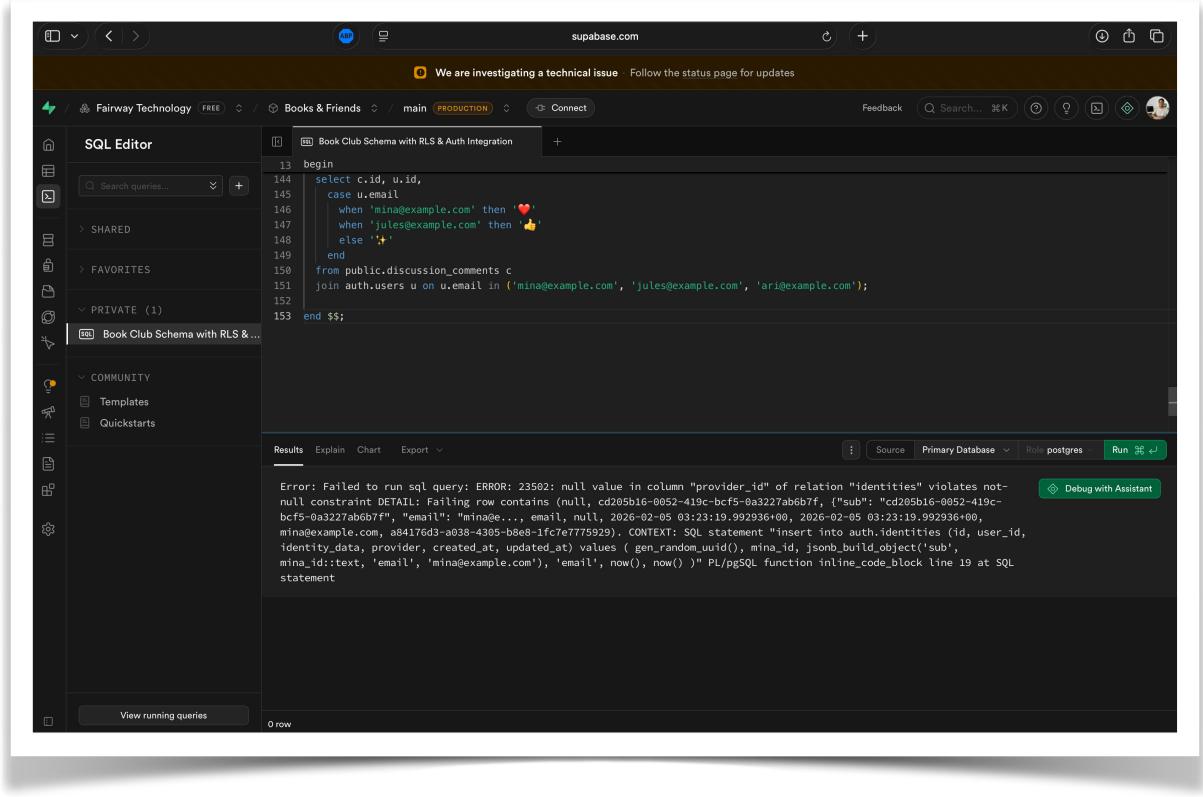
ဒါကြောင့် အခုလို Prompt ပေးလိုက်ပါတယ်။

> *Before implementing next features, tell me what to do on supabase to be ready. I preferred you help me create* ***ready to run code****.*

ဒီလိုပြောလိုက်တဲ့အခါ လိုအပ်တဲ့ Table တွေဖန်တီးဖို့အတွက် ကုဒ်တွေ ချပေးလာပါတယ်။

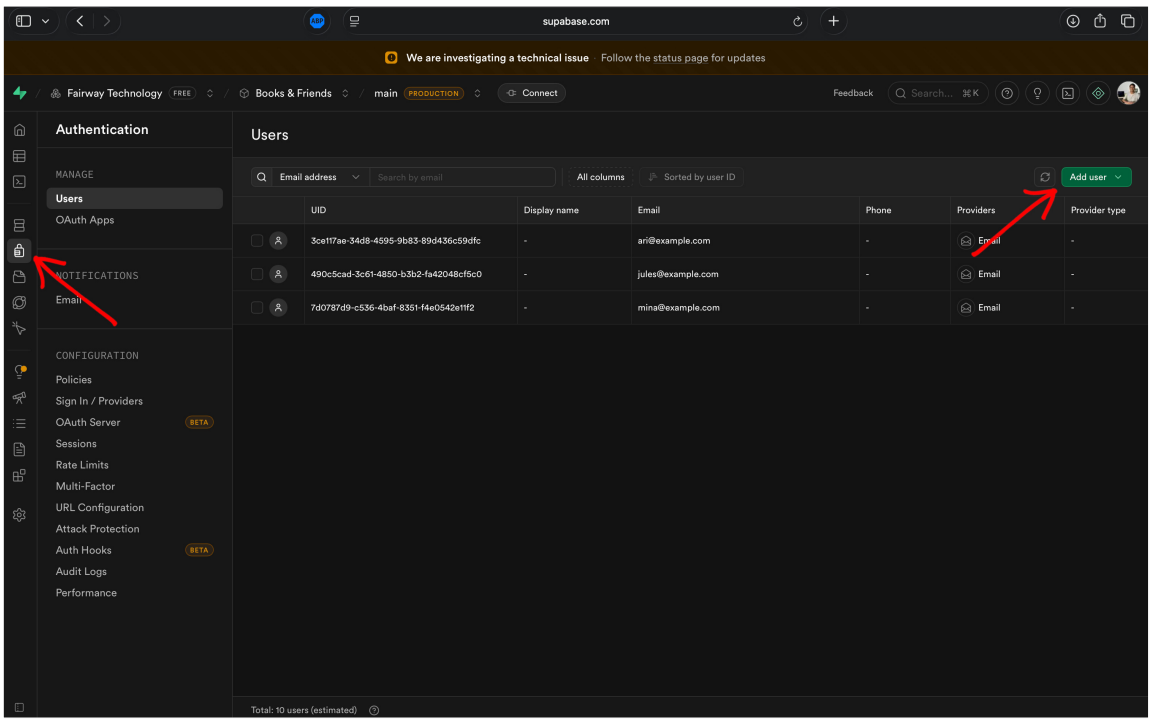


ကူးယူပြီး Supabase မှာ သွား Run လိုက်ပါတယ်။ နောက်တစ်ဆင့်က ပုံမှာပြထားတာက Supabase ဘက်မှာ Error တက်နေတာပါ။



ပြဿနာမရှိပါဘူး။ အဲ့ဒီလိုပဲ တစ်ခါတစ်လေ ဖြစ်တတ်ပါတယ်။ စာရေးသူဆိုရင်တော့ အဲ့ဒီ Error ကို ဖတ်ပြီး ကိုယ်တိုင်အဖြေရှာမှာပါ။ စာဖတ်သူကတော့ အဲ့ဒီ Error ကိုပဲကူးယူပြီး Agent ကိုပေးလိုက်လို့ရပါတယ်။ ပြင်ပေးပါလိမ့်မယ်။

အဓိက ပြဿနာက Supabase Authentication မှာ User အကောင့်တွေဆောက်တဲ့ အလုပ်ကို ကိုယ့်ဘာသာလုပ်မှ အဆင်ပြေမှာ ဖြစ်နေပါတယ်။ Agent ပေးတဲ့ ကုဒ်တွေ Run ရုံနဲ့ အဆင်မပြေတော့ပါဘူး။ ဒါကြောင့် Supabase ဘက်မှာ အခုလို User Account တချို့ ထည့်ပေးလိုက်ပါတယ်။



ပြီးတော့မှသူ့ကို အခုလို အကျိုးအကြောင်းပြောပြပြီး နမူနာဒေတာတချို့ ထည့်တဲ့ကုဒ် တောင်းယူပါတယ်။

> *I have created 3 users on supabase, mina@example.com, jules@example.com and ari@example.com. I need seed data including their display name update. Ready to run script.*

သူပေးတဲ့ ကုဒ်ကိုယူပြီး Supabase ဘက်မှာ Run လိုက်တဲ့အခါ နမူနာဒေတာတွေ အဆင်သင့်ရသွားပြီဖြစ်ပါတယ်။

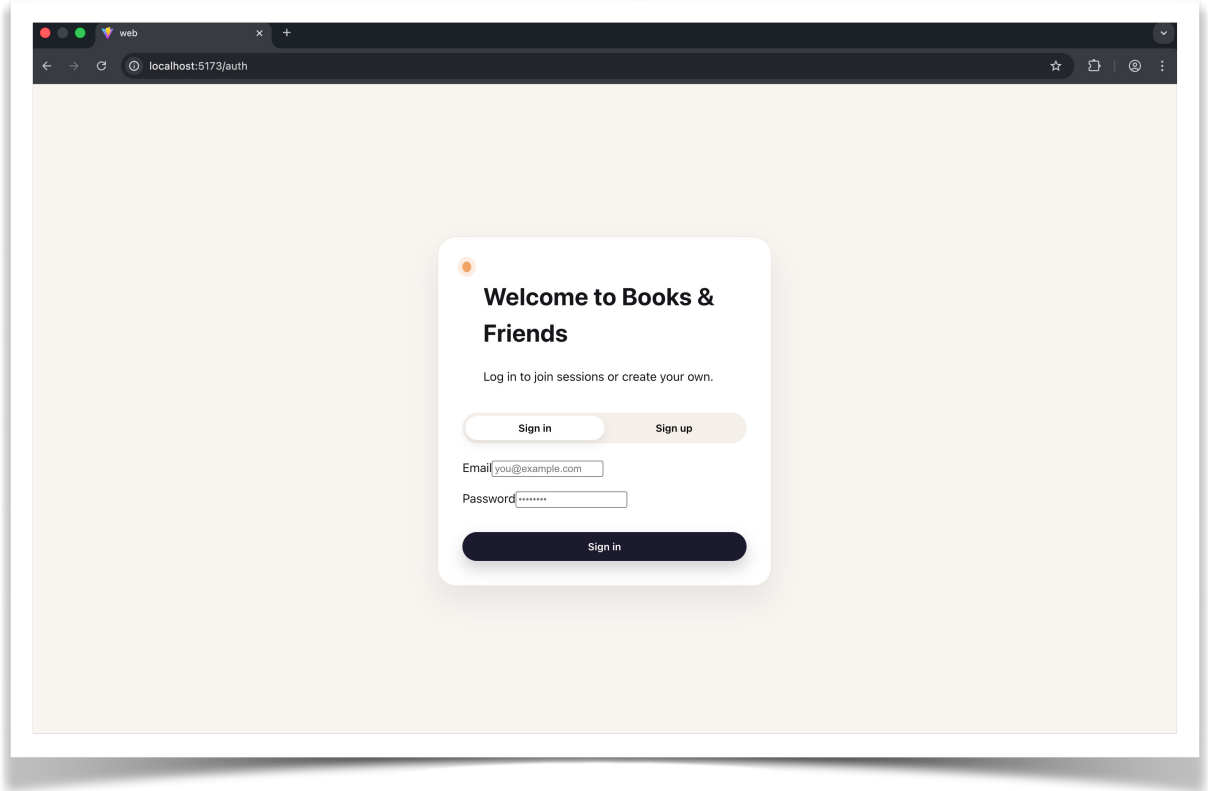
ဒီအထိရသွားတဲ့အခါ Supabase ဘက်မှာ လိုအပ်တာတွေ စုံသွားပါပြီ အသင့်ဖြစ်နေပါ ပြီ။ ဒါကြောင့် ကျန်တာတွေ ဆက်လုပ်လို့ရသွားပါပြီ။ လုပ်ခိုင်းလိုက်ပါတယ်။

> *Now our database is ready. Implement full MVP based on **project specification**.*

ဘာတွေလိုချင်လဲ အသေးစိတ် ထပ်ပြောစရာ မလိုတော့ပါဘူး။ စောစောက ကြိုရေးထားပြီးဖြစ်တဲ့ Detail Spec အတိုင်း လုပ်စရာရှိတာတွေ အကုန်လုပ်ခိုင်းလိုက်တာပါ။ သူလိုအပ်သလောက် အချိန်ယူပြီး ရေးပေးသွားပါလိမ့်မယ်။

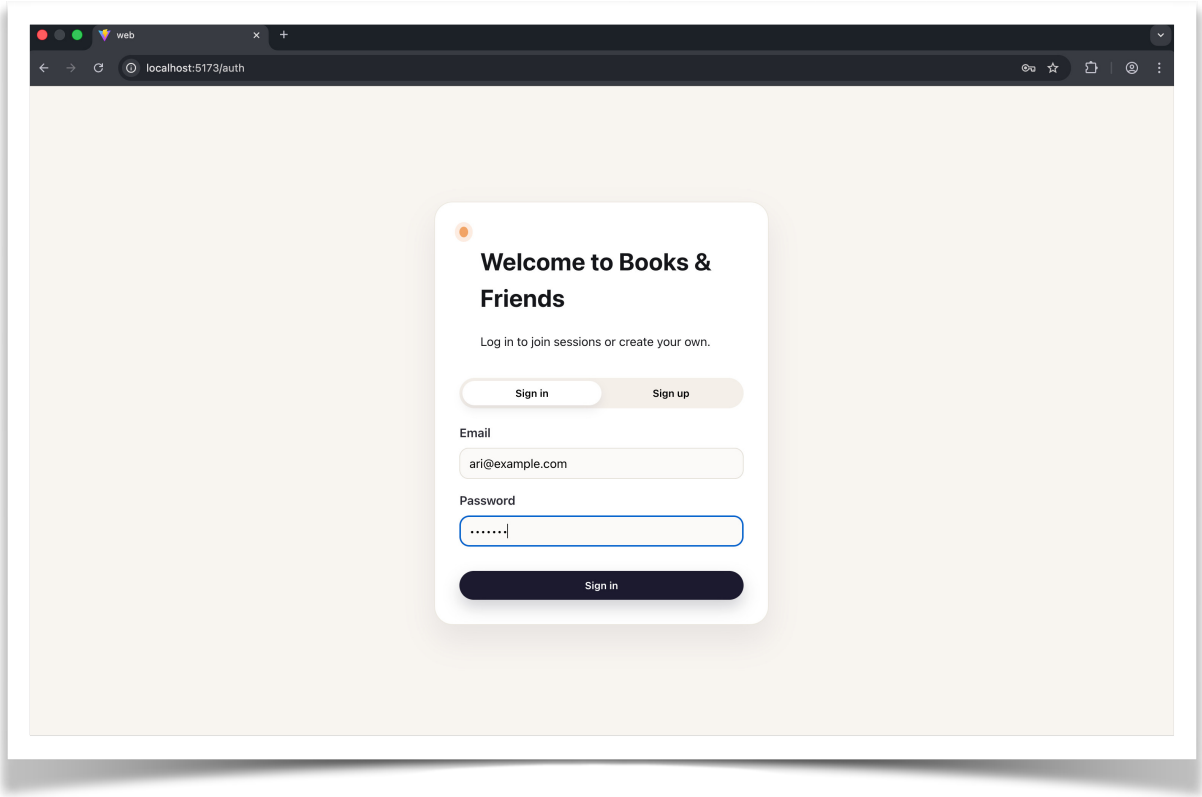
တစ်ချက်တော့ စောင့်ကြည့်ပါ။ ကိုယ့်ဘက်က အတည်ပြု Approve လုပ်ပေးရတာတွေ ရှိတတ်ပါတယ်။ Agent ကတောင်းဆိုလာတဲ့အခါ ကိုယ်က အတည်ပြုပေးဖို့လိုပါတယ်။

ရေးစရာရှိတာတွေ Agent ဆက်လက်ရေးသားသွားပြီးနောက် အခုလို User Account Login ဝင်တာတွေဘာတွေထိ အားလုံးပြည့်စုံသွားပြီဖြစ်တဲ့ ရလဒ်ကို ရလိုက်ပါတယ်။

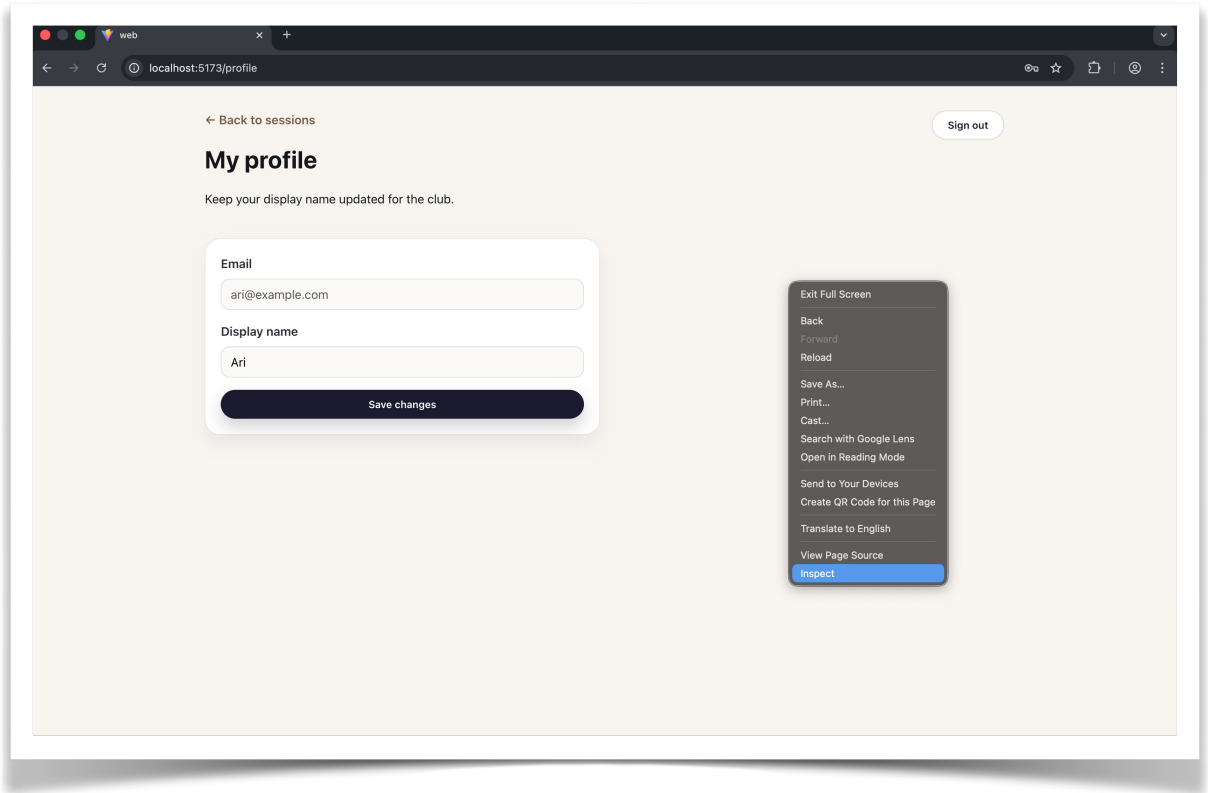


အများအားဖြင့် အဆင်ပြေပေမယ့် Email / Password ရိုက်ထည့်ရတဲ့ Input Box တွေ ကြည့်မကောင်းဖြစ်နေလို့ အဲ့ဒါလေးကို Update လုပ်ပေးပါလို့တော့ နောက်တစ်ကြိမ် ထပ်ပြောလိုက်ရပါတယ်။

ဒီတော့မှ အခုလိုပိုကြည့်ကောင်းတဲ့ ရလဒ်ကို ရရှိသွားပါတယ်။

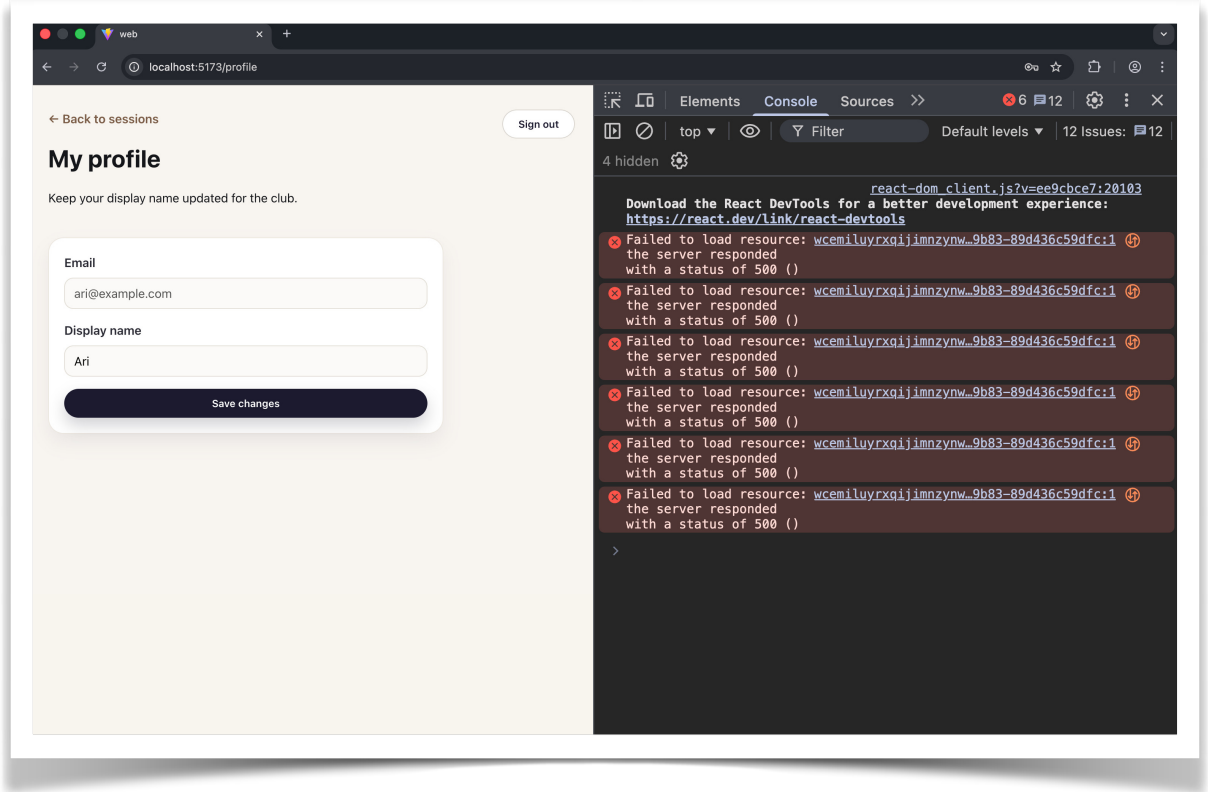


သပ်သပ်ရပ်ရပ်လေး ဖြစ်သွားပါပြီ။ စမ်းကြည့်တဲ့အခါ အလုပ်မလုပ်သေးတာကို တွေ့ရပါတယ်။ ဒါကြောင့် Browser နေရာလွတ်မှာ **Right Click** နှိပ်ပြီး **Inspect** ကိုနှိပ်ပါတယ်။



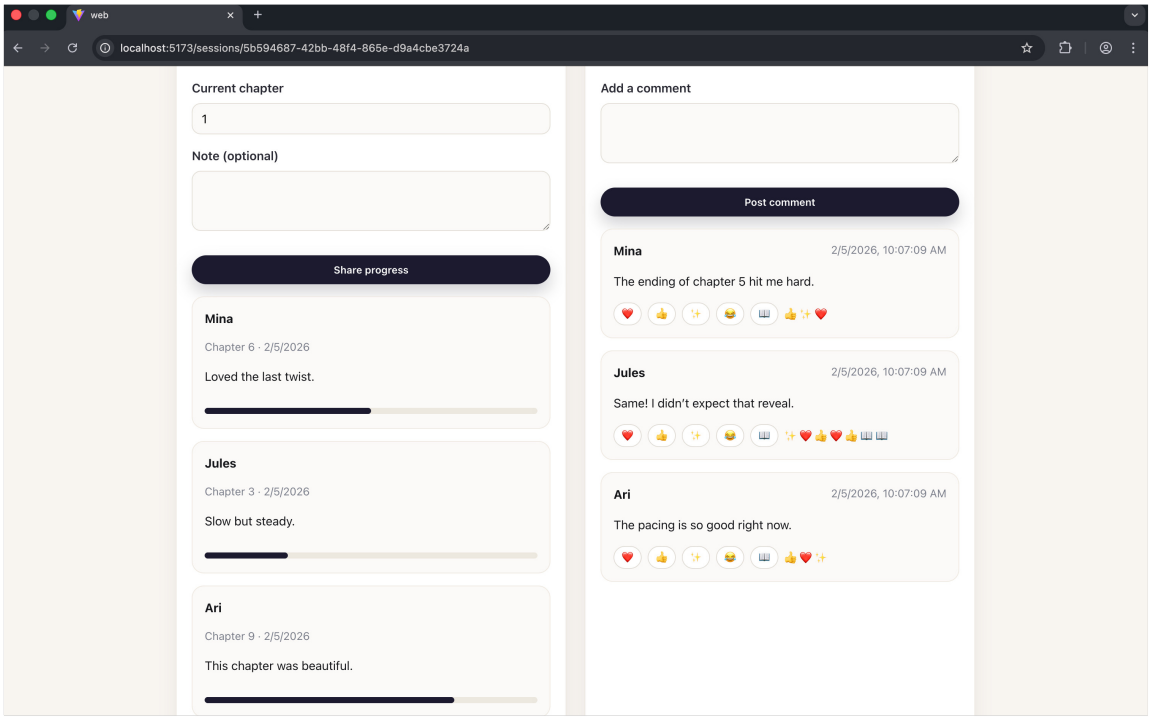
ပေါ်လာတဲ့ Developer Tools ထဲက **Console** ကို ကြည့်လိုက်တဲ့အခါ အခုလို Error တချို့ ရှိနေပါတယ်။

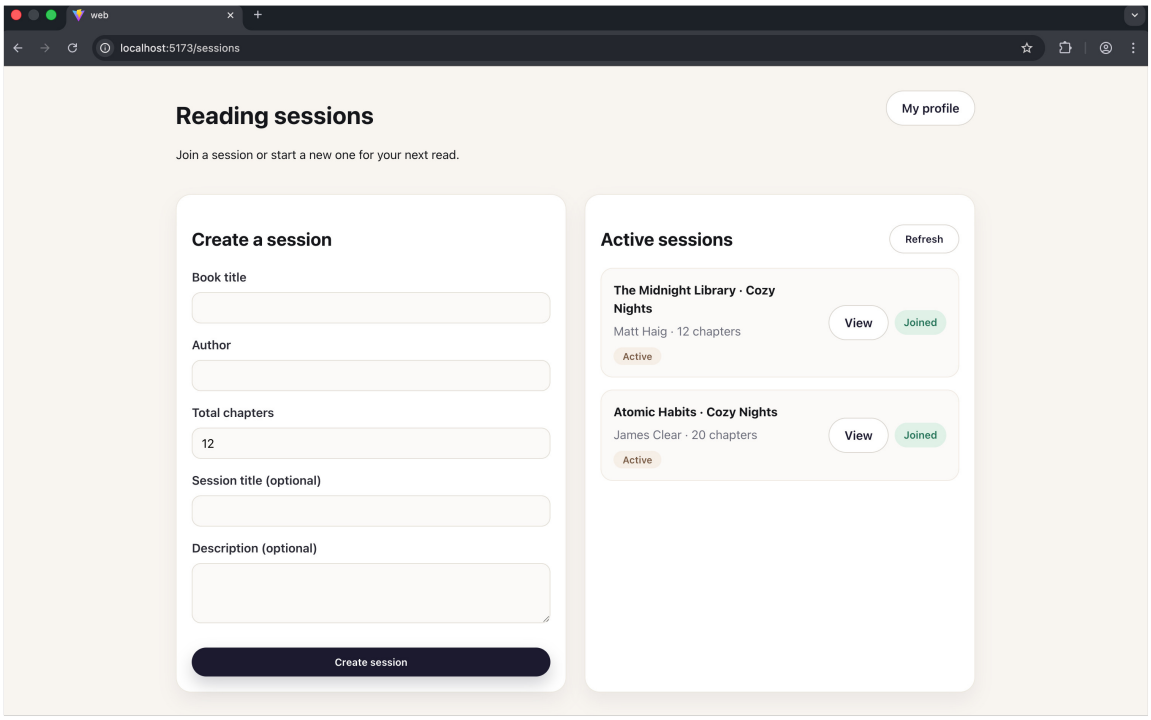
စာရေးသူကတော့ အဲဒီ Error တွေကို လိုက်ဖတ်ပြီး ဖြစ်နိုင်ခြေအဖြေကို ကိုယ့်ဘာသာ အရင်စဉ်းစားမှာပါ။ စာဖတ်သူကတော့ ထုံးစံအတိုင်း အဲဒီ Error ကို ကူးယူပြီး Agent ကို ပြလိုက်လို့ ရပါတယ်။ Agent က အဖြေရှာပေးပါတယ်။



Supabase ဘက်မှာ Permission တချို့ ပြင်ဖို့လိုနေတာပါ။ အဲ့ဒီလို ပြင်ဖို့အတွက် Agent ရေးပေးတဲ့ ကုဒ်တွေကိုပဲ ကူးယူပြီး Supabase ဘက်မှာ Run လိုက်တဲ့အခါ အဆင်ပြေသွားပါတယ်။

အားလုံးအလုပ်လုပ်သွားပါပြီ။ ကိုယ်လိုချင်တဲ့အတိုင်း User တွေ စာဖတ် Session ဖွင့်လို့ရသွားပါပြီ။ တခြား User တွေ Join ပြီး စာအုပ်တွေ အတူတူဖတ်လို့ရပါပြီ။ Discussion နဲ့ တိုင်ပင်လို့ ရပါပြီ။ Reaction တွေ ပေးလို့ရပါပြီ။ Spec မှာ ပြောထားတဲ့အတိုင်း လိုချင်တာ အားလုံးပါတဲ့ ရလဒ်တစ်ခုကို အောင်မြင်စွာ ရရှိလိုက်တာပါ။


web
localhost:5173/sessions/5b594687-42bb-48f4-865e-d9a4cbe3724a
Current chapter
1
Note (optional)
Share progress
Mina
Chapter 6 · 2/5/2026
Loved the last twist.
Jules
Chapter 3 · 2/5/2026
Slow but steady.
Ari
Chapter 9 · 2/5/2026
This chapter was beautiful.
Add a comment
Post comment
Mina
2/5/2026, 10:07:09 AM
The ending of chapter 5 hit me hard.
Jules
2/5/2026, 10:07:09 AM
Same! I didn't expect that reveal.
Ari
2/5/2026, 10:07:09 AM
The pacing is so good right now.


web
localhost:5173/sessions
Reading sessions
Join a session or start a new one for your next read.
My profile
Create a session
Book title
Author
Total chapters
12
Session title (optional)
Description (optional)
Create session
Active sessions
Refresh
The Midnight Library · Cozy Nights
Matt Haig · 12 chapters
Active
View
Joined
Atomic Habits · Cozy Nights
James Clear · 20 chapters
Active
View
Joined

ပထမ Version ပြည့်စုံပြီဖြစ်လို့ နောက် Version ကို မသွားခင် Spec ကို အခုလို Update လုပ်ခိုင်းလိုက်ပါတယ်။

*Add our current **supabase setup to project spec** for future context.*

ဒီတော့မှ Agent က သူဖန်တီးခဲ့တဲ့ Table တွေရဲ့ဖွဲ့စည်းပုံအချက်အလက်တွေကို Spec ဖိုင်ထဲမှာ Update လုပ်ပေးသွားလို့ နောင်သိချင်ရင် Supabase ကို သွားကြည့်နေစရာ မလိုတော့ပါဘူး။ Spec ဖိုင်မှာ ကြည့်လိုက်ရုံပါပဲ။

Next Version

နောက် Version မှာ လုပ်မယ်ဆိုရင် လုပ်လို့ရတာတွေ အများကြီးရှိပေမဲ့ User တွေရဲ့ ပရိုဖိုင်ပုံတင်လို့ရတဲ့ လုပ်ဆောင်ချက်ကို လုပ်လိုက်ချင်ပါတယ်။ ဒီအတွက် လက်ရှိ သုံးလက်စ Agent Session ကို ဆက်မသုံးတော့ဘဲ Session အသစ် ဖွင့်လိုက်မှာ ဖြစ်ပါတယ်။

ဒီနေရာမှာ မဖြစ်မနေလိုအပ်လို့မဟုတ်ပါဘူး။ လိုအပ်တဲ့အခြေအနေတွေ ရှိတတ်ပါတယ်။ ဒါလေးက အရေးကြီးပါတယ်။

- AI က အများတစ်ခုလုပ်ပြီးနောက် ဆက်တိုက် အများတွေ ဆက်လုပ်နေတယ်။
- မဟုတ်ဘဲနဲ့ ဟုတ်ပါတယ်ပြောတာမျိုး အပါအဝင် Hallucination ဝင်ပြီး ယောင်ဝါးဝါး ဖြစ်နေတယ်။

- တချို့ မှတ်မိသင့်တာတွေကို AI က မမှတ်မိဘူး ဖြစ်နေတယ်။ တချို့ သူလုပ်ထားတာ သူဘာသာ ပြန်မေ့နေတယ်။

စသည်ဖြင့် အလုပ်လုပ်တာ ကြာလာတဲ့အခါ ဖြစ်တတ်တဲ့ သဘောတွေ ရှိပါတယ်။ အကြောင်းရင်း (၃) ခု ရှိပါတယ်။

- Exponential Error
- Context Cutoff
- Misalignment

ဟိုးရှေ့မှာ Generative AI တွေရဲ့ အလုပ်လုပ်ပုံကို ရှင်းပြခဲ့ပါတယ်။ ဘယ်လောက်ပဲ Smart ဖြစ်တယ်ထင်ရထင်ရ သူ့ရဲ့ တကယ့်အလုပ်လုပ်ပုံက နောက်လာမယ့်စာလုံးဘာဖြစ်နိုင်လဲ ခန့်မှန်းပြီး ထုတ်ပေးတာပါပဲ။ တစ်ခုမှားသွားရင် နောက်ဆက်တွဲတွေကလည်း မှားနိုင်ပါတယ်။ ရှေ့တစ်နေရာက အဲ့ဒီအမှားလေး ရှိနေသ၍ နောက်ပိုင်း ဘယ်လိုမှ အမှန်ပြင်လို့မရတော့တာ ဖြစ်နိုင်ပါတယ်။ ဒီလိုအခြေအနေမျိုးမှာ AI နဲ့ ဒီလောက်လေးတောင် မလုပ်နိုင်ဘူးလားဆိုပြီး ရန်ပြန်ဖြစ်မနေဘဲ Session သစ်ကို ဖွင့်လိုက်ဖို့ လိုတာပါ။

နောက်တစ်ချက်ကတော့ Context အကန့်အသတ်ပါ။ အပြန်အလှန်ပြောတဲ့စကားတွေ၊ ရေးပေးတဲ့ကုဒ်တွေ Session တစ်ခုထဲမှာ များလာတဲ့အခါ လက်ခံနိုင်တဲ့ Context ပမာဏကို ကျော်သွားတာ ဖြစ်နိုင်ပါတယ်။ ဒီလိုအခြေအနေမှာ AI က ရှေ့ပိုင်းက Context အဟောင်းတွေကို ချန်ပြီး အလုပ်လုပ်ပေးသွားမှာပါ။ ဒီတော့ တချို့

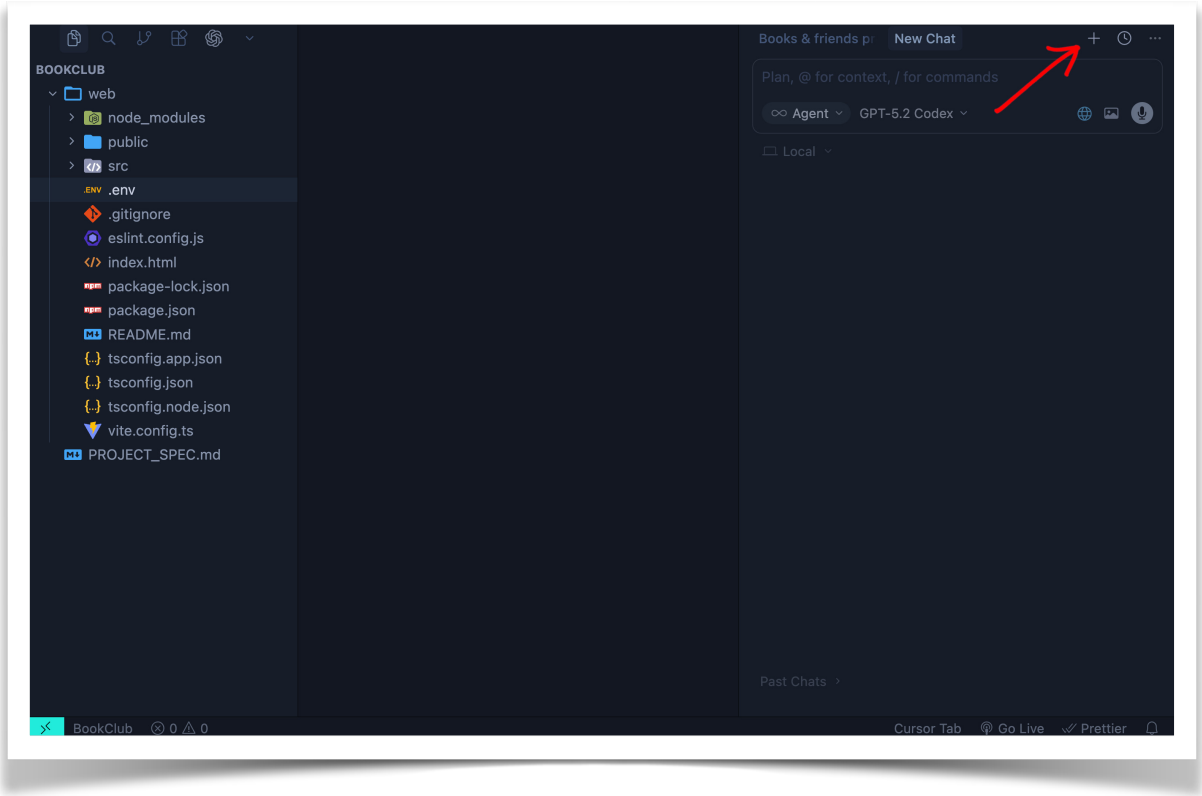
အကြောင်းအရာတွေ Context ထဲမှာ မပါတော့လို့ သူကိုယ်တိုင် သူလုပ်ထားတာကို မသိတော့တာမျိုးတွေဟာ ဖြစ်နိုင်ပါတယ်။

Agent က လက်ရှိ Session မှာ Context ပမာဏ ဘယ်လောက်ထိသုံးထားလဲ ဆိုတာကို အောက်ခြေနားလေးမှာ Progress Percentage လေးနဲ့ ပြကြလေ့ရှိပါတယ်။

နောက်ဖြစ်နိုင်တာကတော့၊ တစ်ခါတစ်လေ တကယ်လည်း မမှားဘဲနဲ့ ကိုယ့်ရဲ့ စမ်းပုံစမ်းနည်းမှားပြီး AI ကို မှားနေတယ်လို့ ပြောမိတာမျိုးတွေ ရှိတတ်ပါတယ်။ အဲ့ဒီမှာ AI က မှန်ရဲ့သားနဲ့ မှားတယ်ထင်ပြီး လျှောက်လုပ်တော့ ပိုရှုပ်ကုန်တာမျိုးတွေ ရှိတတ်ပါတယ်။

တစ်ခါတစ်လေ AI ရေးပေးတဲ့ ကုဒ်ကြောင့် မဟုတ်ဘဲ ကိုယ့်စက်ကြောင့်၊ ဒါမှမဟုတ် Setting တခုခုကြောင့် အလုပ်မလုပ်တာကို AI ကို Complain လုပ်မိရင် AI က ကုဒ်မှားတယ်ထင်ပြီး လျှောက်လုပ်သွားတာတွေ ရှိတတ်ပါတယ်။ နောက်ပိုင်း သိလို့ အမှန်ပြင်လိုက်နိုင်ရင်တောင် အဲ့ဒီလို လွဲချော်ထားတဲ့ အလွဲအချော်ရှိနေရင်၊ အဲ့ဒီအလွဲအချော်ကြီးကို AI က ပြန်ကိုးကားပြီး ပြဿနာတတ်နိုင်ပါသေးတယ်။

ဒါမျိုး ပြဿနာတွေကြောင့် Session အသစ်ဖွင့်ဖို့ မဖြစ်မနေလိုတာ ရှိလာနိုင်ပါတယ်။ အကျဉ်းချုပ် မှတ်ထားသင့်တာကတော့ လိုသည်ဖြစ်စေ မလိုသည်ဖြစ်စေ၊ တစ်ပိုင်းပြီးရင် နောက်တစ်ပိုင်းကို Session အသစ်နဲ့ စသင့်တယ်လို့ မှတ်ထားနိုင်ပါတယ်။

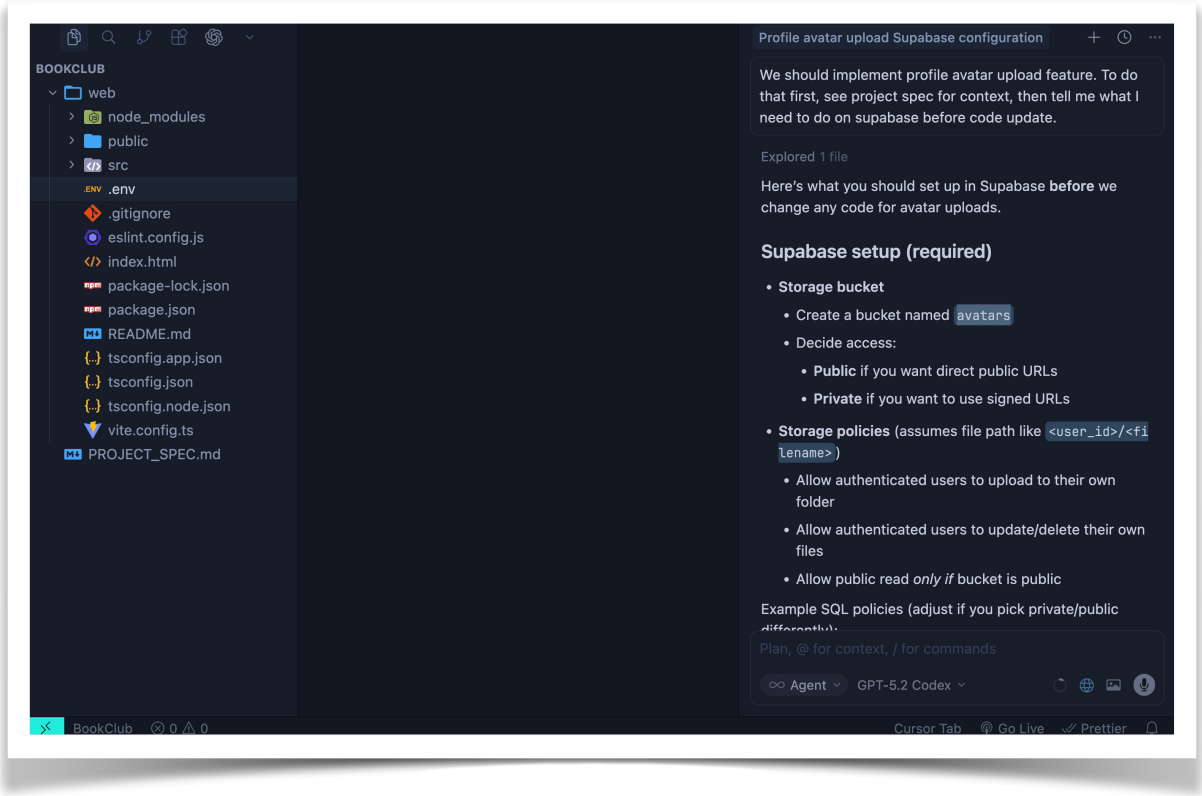


Session အသစ်မှာ ဘာ Context မှမရှိသေးပါဘူး။ ဘာ Context မှ မရှိသေးဘဲနဲ့ AI က ကိုယ့်ပရောဂျက်အကြောင်း ဘာမှ မသိသေးပါဘူး။ ဒါကြောင့် ခိုင်းချင်တာတွေ တန်းမခိုင်းသင့်ပါဘူး။ ပုံမှန်အားဖြင့် အရင်ဆုံး လက်ရှိရေးထားတဲ့ ကုဒ်တွေကို လေ့လာခိုင်းရပါတယ်။ လေ့လာပြီးပြီ ဆိုတော့မှ ခိုင်းချင်တာကို ခိုင်းရပါတယ်။

ပုံမှန်အားဖြင့် AI ကို ရေးထားလက်စကုဒ်တွေ ဖတ်ကြည့်ခိုင်းလို့ ရပါတယ်။ အခုတော့ Spec အပြည့်အစုံ ရှိတဲ့အတွက် ရေးထားတဲ့ကုဒ်တွေ တစ်ဖိုင်ချင်းဖတ်ပြီး လေ့လာခိုင်းနေစရာတောင် မလိုတော့ပါဘူး။ အခုလို ပြောလိုက်ရုံပါပဲ။

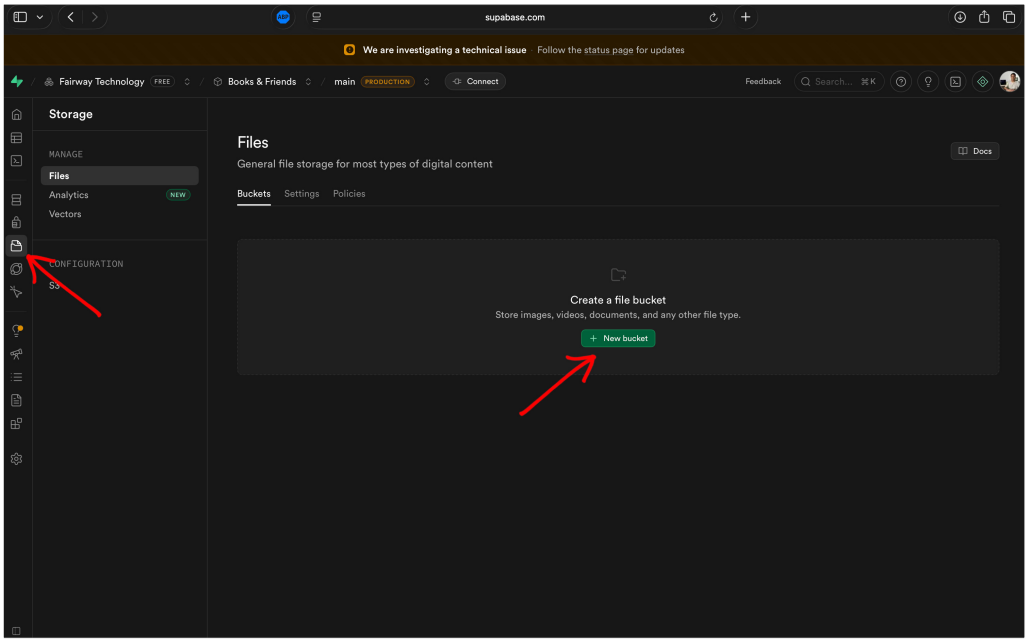
> *Implement profile avatar upload feature. To do that first,* ***see project spec*** *for context, then* ***tell me what I need to do*** *on supabase before code update.*

Context ရအောင် Spec ကို ဖတ်ခိုင်းလိုက်ပြီး Supabase ဘက်မှာ ဘာလုပ်ပေးရမလဲ မေးလိုက်တာပါ။ သူက Step by Step လုပ်ရမှာတွေ အခုလို ပြောပြလာပါတယ်။

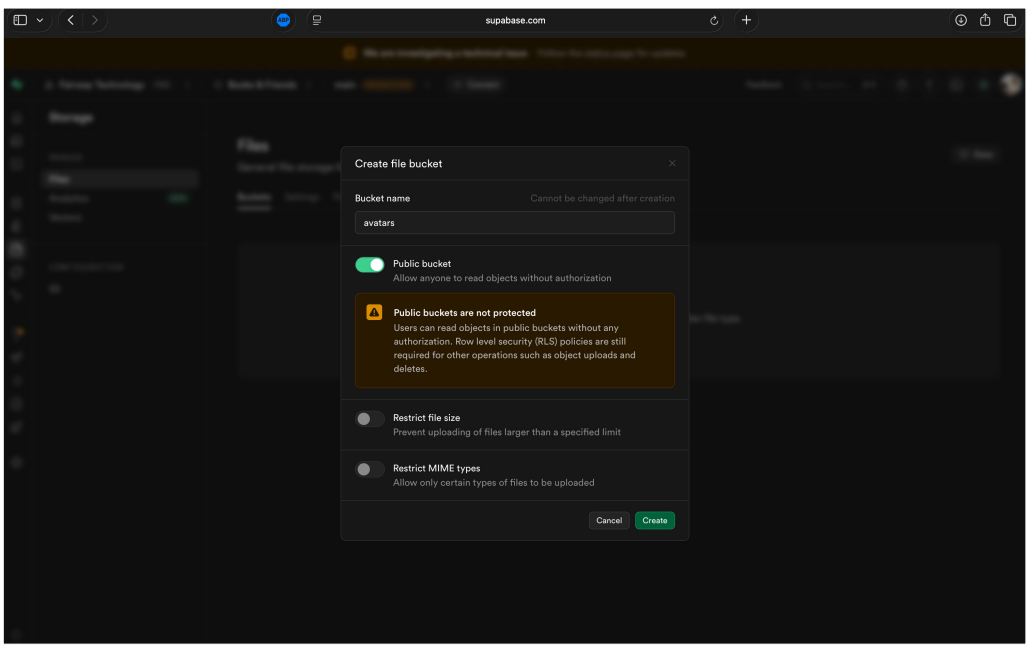


Supabase ရဲ့ Storage Section ကိုသွားပြီး ဖိုင်တွေသိမ်းလို့ရအောင် `avatars` အမည်နဲ့ Bucket အသစ်တစ်ခု တည်ဆောက်ပေးပါလို့ ပြောထားပါတယ်။

ဒါကြောင့် ပြောတဲ့အတိုင်းသွားပြီး တည်ဆောက်လိုက်ပါတယ်။

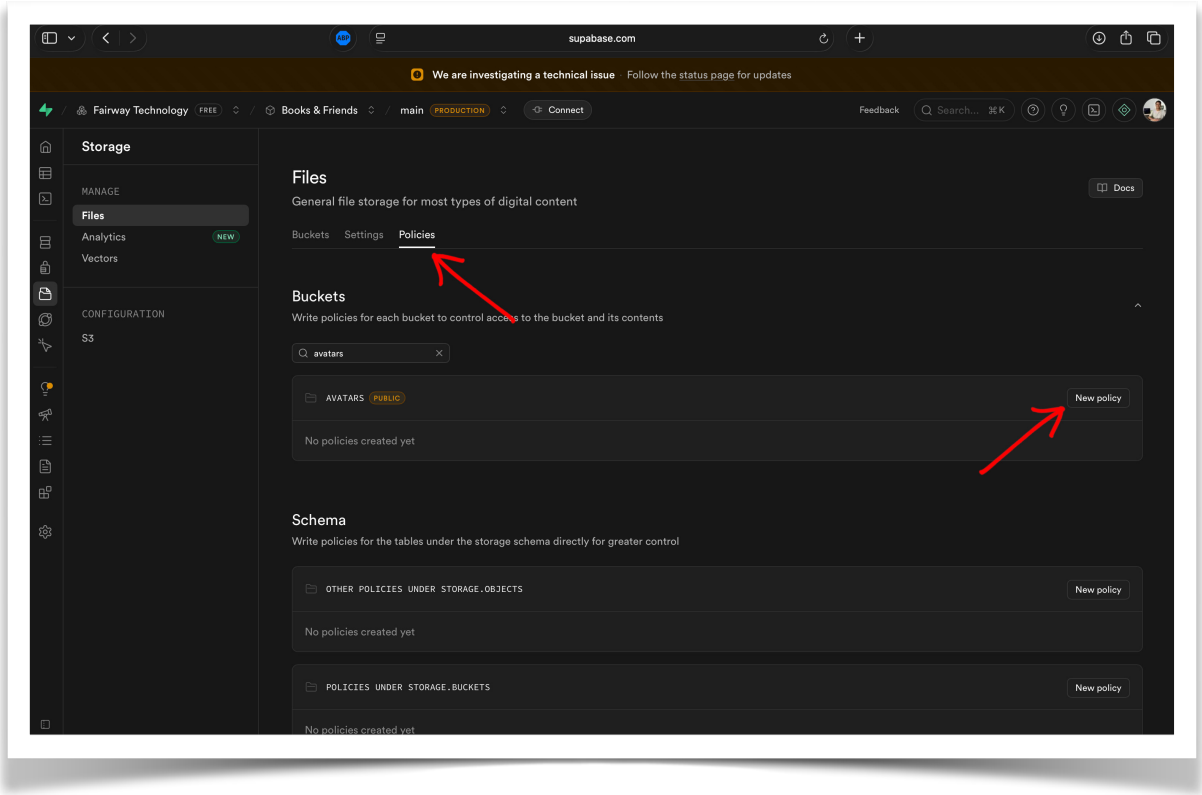


ပရိုဖိုင်ပုံတွေက လုံခြုံရေးပြဿနာ သိပ်မရှိလို့ Public Bucket အနေနဲ့ပဲ တည်ဆောက်လိုက်ပါတယ်။ မည်သူမဆို ဝင်ကြည့်လို့ရတဲ့နေရာတစ်ခု ဖြစ်သွားတာပါ။



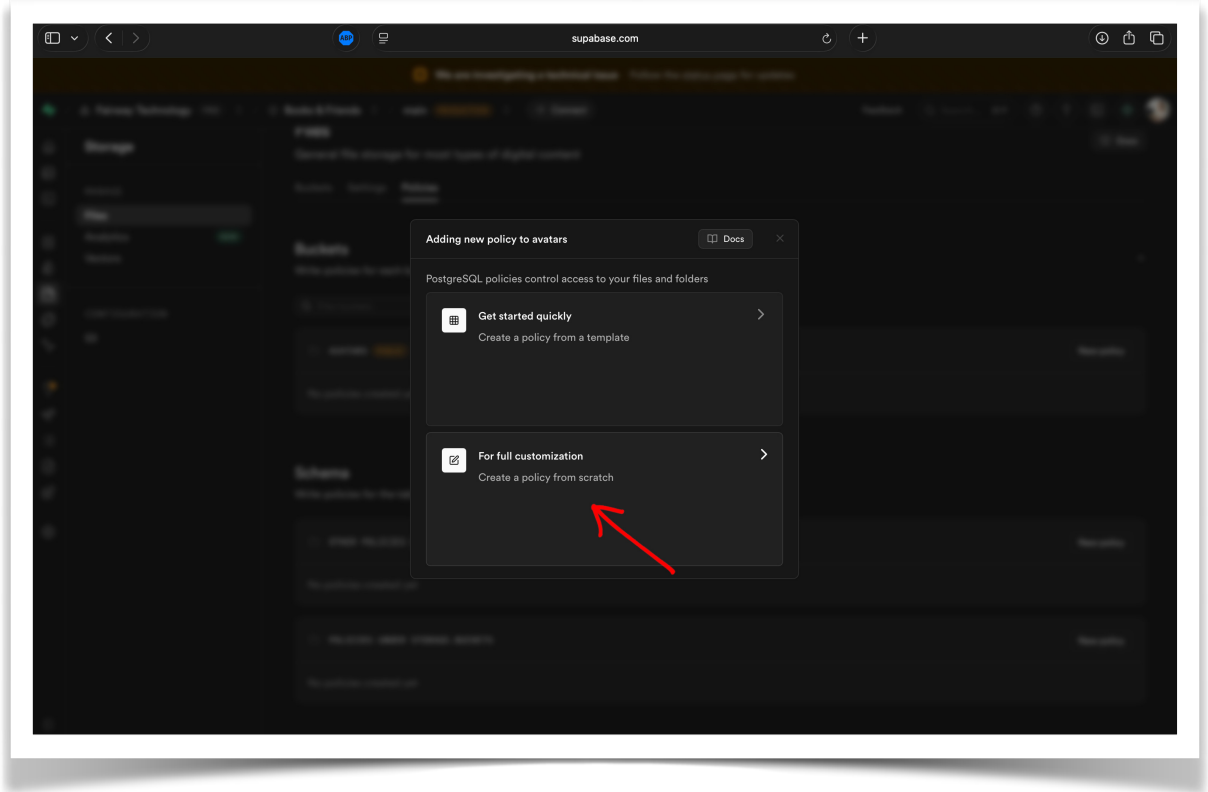
အကယ်၍ အပြင်လူတွေမသိစေချင်တဲ့ ဖိုင်တွေဆိုရင်တော့ Private ထားရမှာပါ။

နောက်တစ်ဆင့်အနေနဲ့ အဲ့ဒီ Bucket ထဲကဖိုင်တွေကို ဘယ်သူတွေ ယူခွင့်ရှိလဲ၊ ဘယ်သူတွေပြင်ခွင့်ရှိလဲ Policy တွေ သတ်မှတ်ပေးဖို့လိုပါတယ်။



ဒီပေါ်လစီတွေသတ်မှတ်ဖို့ AI ကိုပဲ ကုဒ်တွေ ရေးခိုင်းလို့လည်း ရပါတယ်။ သူရေးပေးတဲ့ကုဒ်ကို Supabase မှာ Run လိုက်ရုံပါပဲ။ နမူနာမှာ ကိုယ့်ဘာသာ ပေးလိုက်မိတဲ့အတွက် ပေးခဲ့တဲ့ အဆင့်တွေကို ထည့်သွင်း ဖော်ပြလိုက်ပါတယ်။

New Policy ခလုတ်ကိုနှိပ်ပြီးနောက် နောက်တစ်ဆင့်မှာ **Full Customization** ကို ရွေးပါတယ်။

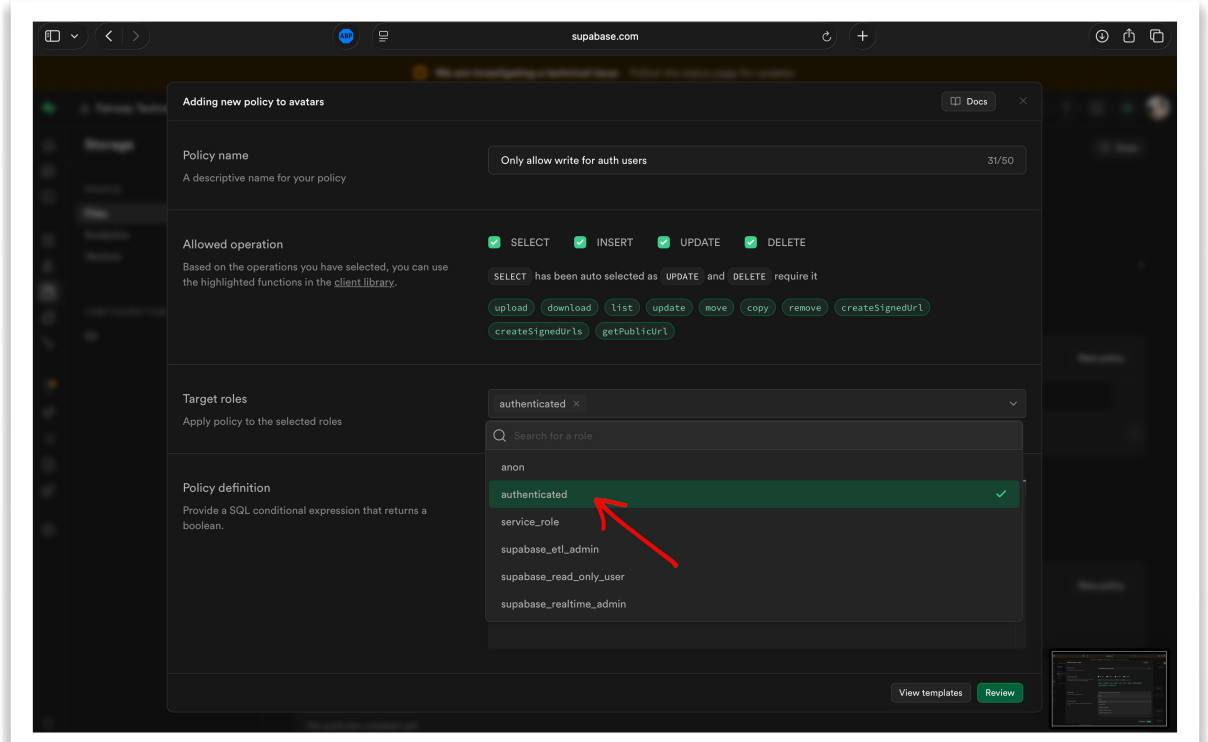


ပြီးတဲ့အခါ Policy Name ကို “**Allow everyone to read file**” လို့ပေးပြီး `SELECT` ကို ရွေးပါတယ်။ ပြီးတဲ့အခါ **Review** ခလုတ်ပြီးနှိပ်ပြီး **Save** လုပ်ပေးလိုက်ပါတယ်။

နောက်တစ်ဆင့်အနေနဲ့ နောက်ထပ် Policy တစ်ခုထပ်ထည့်ပြီး -

Policy Name ကို “**Only allow write for auth users**” လို့ပေးပါတယ်။ `SELECT, INSERT, UPDATE, DELETE` အားလုံးကိုရွေးပါတယ်။ ပြီးတဲ့အခါ Target roles အဖြစ် `authenticated` ကို ရွေးပါတယ်။ **Review, Save** လုပ်လိုက်ပါတယ်။

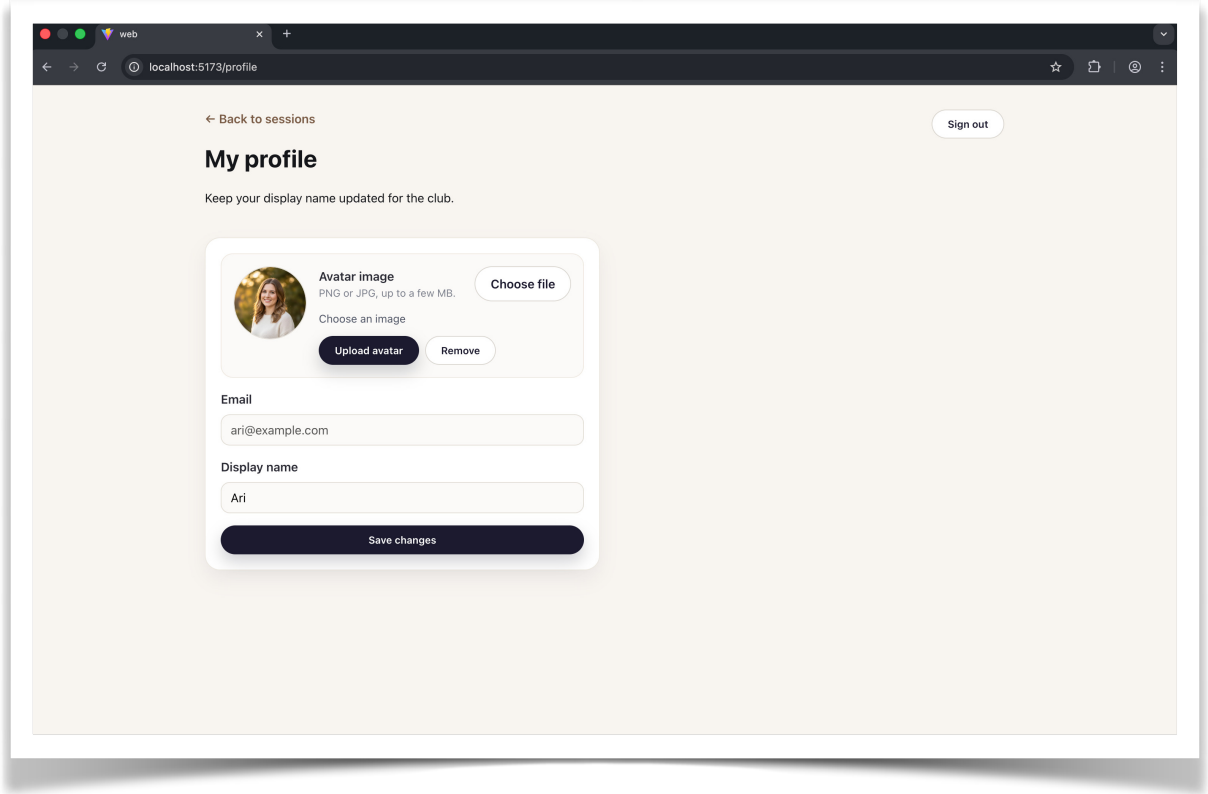
ဒါဘာလုပ်လိုက်တာလဲဆိုတော့ အရင်ဆုံး လူတိုင်းကို `SELECT` လုပ်ခွင့်ပေးပြီး၊ Login ဝင်ထားသူမှသာ ကျန်လုပ်ဆောင်ချက်တွေကိုပါ လုပ်ခွင့်ပေးလိုက်တာပါ။



SELECT, INSERT, UPDATE, DELETE စတဲ့ အသုံးအနှုန်းတွေဟာ SQL Language ကလာတဲ့ ကုဒ်တွေဖြစ်ပြီး Coding ရတဲ့သူတွေကတော့ သိကြပါတယ်။ Coding မရလို့ မသိရင်လည်း အကျဉ်းချုပ်အားဖြင့် **SELECT** ဆိုတာ လိုချင်တဲ့ အချက်အလက်တွေကို ရယူတာလို့ မှတ်နိုင်ပါတယ်။ **INSERT** က အသစ်ထည့်တာဖြစ်ပြီး၊ **UPDATE** တို့ **DELETE** တို့ကတော့ ပြန်ပြင်တာနဲ့ ပယ်ဖျက်တာတွေ ဖြစ်ကြပါတယ်။

Supabase ဘက်က အားလုံးအသင့်ဖြစ်သွားပြီမို့လို့ Agent ကို Profile Avatar ထည့်လို့ရတဲ့ Feature ရေးခိုင်းလိုက်တဲ့အခါ ဘာမှအများကြီးထပ်ပြောစရာမလိုဘဲ လိုအပ်တာအခုလိုရရှိသွားပါတယ်။

Upload ခလုတ်ကိုနှိပ်ပြီး ပရိုဖိုင်ပုံတွေ ထည့်လို့ရသွားတာပါ။



ဒီနည်းနဲ့ Supabase ကို Database နဲ့ ဒေတာတွေ အတွက်ပါမဟုတ်ဘဲ ဖိုင်တွေအတွက်ပါ အသုံးပြုလို့ရတာပဲ ဖြစ်ပါတယ်။

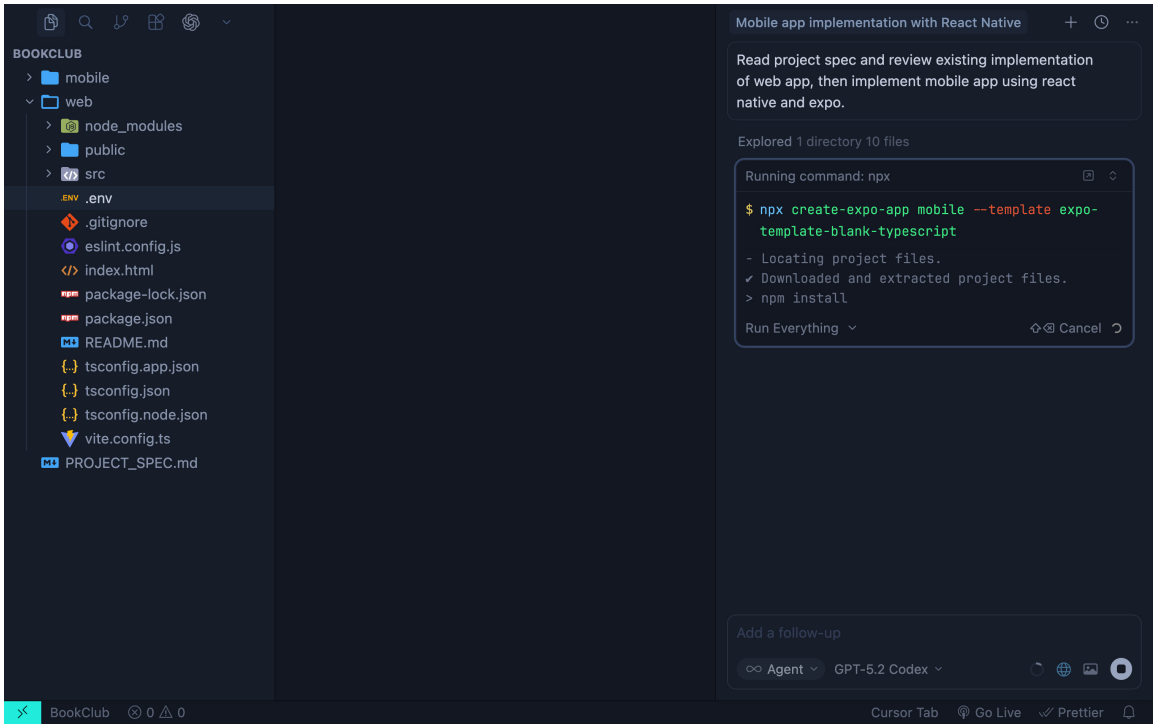
နောက်တစ်ဆင့်မှာ Mobile App လေး ဆက်လုပ်ကြည့်ကြပါမယ်။

Mobile App

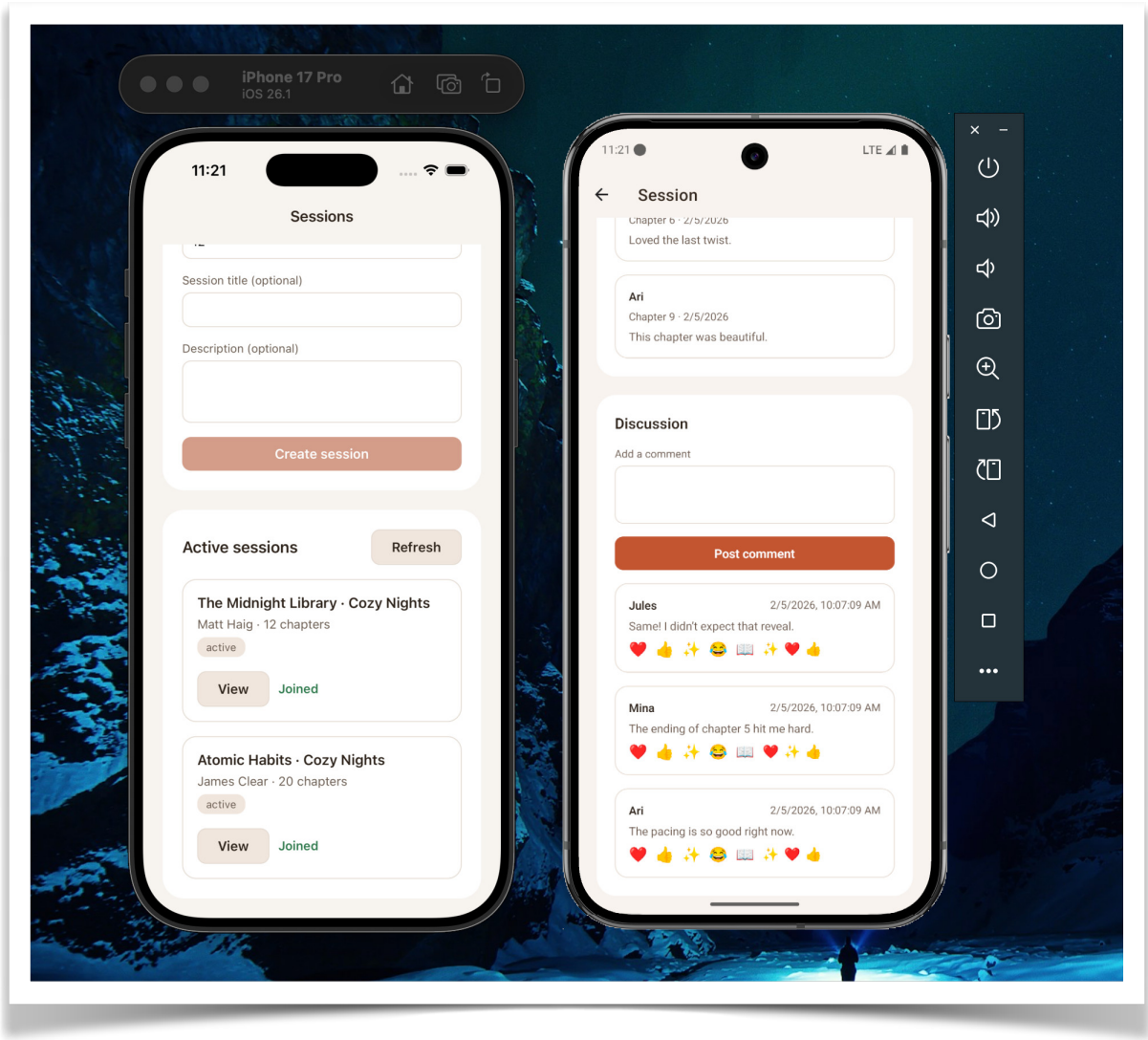
Spec လည်းရှိတယ်၊ Web App လည်း ပထမတစ်ဆင့် စိတ်တိုင်းကျရသွားပြီဖြစ်လို့ Mobile App က တော်တော်လွယ်သွားပါပြီ။ အခုလို Prompt တိုလေးတစ်ခု ပေးလိုက်ရုံနဲ့

လုံးဝအပြီးအစီး တစ်ချက်တည်းနဲ့ ရသွားတာပါ။ တကယ့်ကို အံ့အားသင့်စရာ စွမ်းဆောင်ရည်တွေလို့ ပြောလို့ရပါတယ်။ စာရေးသူကိုယ်တိုင် အံ့သြနေရဆဲပါပဲ။

> ***Read project spec** and **review existing implementation of web app**, then **implement mobile app** using react native and expo*



Spec ကိုကြည့်၊ Web App ဘယ်လိုလုပ်ထားလဲကြည့်၊ ပြီးရင် React Native နဲ့ Expo ကို သုံးပြီး Mobile App လုပ်လိုက်တော့လို့ပဲ ပြောထားတာပါ။ တကယ်ကို ဘာဆိုဘာမှ ထပ်ပြောစရာ မလိုဘဲ အခုလို ရသွားတာပါ။



ပုံမှာပြထားသလို စမ်းလို့ရချင်ရင်တော့ စာဖတ်သူရဲ့ စက်ထဲမှာ လိုအပ်တာအားလုံး အပြည့်အစုံရှိဖို့ လိုပါတယ်။

- Android Studio + Android Emulator
- Xcode + iOS Simulator

Xcode နဲ့ iOS Simulator ကတော့ macOS မှာပဲ ထည့်လို့ရမှာပါ။ Android Studio နဲ့ Emulator ကတော့ Windows မှာလည်း ထည့်လို့ရပါတယ်။

- Android Studio - https://developer.android.com/studio
- Xcode - https://www.xcodes.app/

Run ပုံ Run နည်းကတော့ Cursor မှာပဲ Terminal ဖွင့်လိုက်ပါ။ ပုံမှာ မျှားပြထားတဲ့ ခလုတ်က Terminal မှာ ပြချင်တာများလို့ အပြည့်နေရာယူချဲ့ပြလိုက်တာပါ။

ပြီးတဲ့အခါ အခုလို Run လိုက်ရင် ရပါပြီ။

```
cd mobile
npm start
```

`cd mobile` ဆိုတာ Mobile App ကုဒ်ဖိုဒါထဲကို ဝင်လိုက်တာပါ။ `npm start` ကတော့ ပရောဂျက်ကို Run လိုက်တာဖြစ်ပါတယ်။ ပြီးခဲ့တဲ့ပုံမှာပြထားသလို QR Code နဲ့ အချက်အလက်တချို့ ပေါ်လာပါလိမ့်မယ်။

ကိုယ့်ရဲ့ ဖုန်းထဲမှာ **Expo Go** App ကို Install လုပ်ထားပြီး အဲ့ဒီ App နဲ့ QR ကုဒ်ကို Scan လုပ်လိုက်ပြီး စမ်းလို့ရပါတယ်။ ကွန်ပျူတာနဲ့ ဖုန်းကတော့ Wifi တစ်ခုတည်းမှာ အတူတူ ရှိရပါတယ်။

Expo Go App ကို Google Play Store နဲ့ Apple App Store တို့မှာ ရှာဖွေထည့်သွင်းနိုင်ပါတယ်။ ဒါမှမဟုတ် အောက်ကလင့်ကို သွားလိုက်ရင်လည်း ရပါတယ်။

- Expo Go App - https://expo.dev/go

သူက Expo နဲ့ ဖန်တီးထားတဲ့ Mobile App တွေကို Run ပေးနိုင်တဲ့နည်းပညာတမျိုးပါ။

Android Emulator တွေ iOS Simulator တွေ အကုန်စုံအောင် Install လုပ်ထားတာ ရှိရင် Keyboard က `i` ကို နှိပ်ပြီး iOS Simulator နဲ့ ဖွင့်လို့ရပါတယ်။ a ကို နှိပ်ပြီး Android Emulator နဲ့ ဖွင့်ပြီး စမ်းကြည့်လို့ရပါတယ်။

Burmese Language

လက်ရှိလုပ်ပြထားတဲ့ နမူနာမှာ အားလုံးက အင်္ဂလိပ်လို ဖြစ်နေပါတယ်။ Prompt ကို ဘာကြောင့် အင်္ဂလိပ်လိုပေးသလဲဆိုတာ အထက်မှာတစ်ကြိမ် ပြောဖြစ်ခဲ့ပါတယ်။ AI က မြန်မာလိုလည်း နားလည်ပေမဲ့ အင်္ဂလိပ်လို ပိုကျွမ်းလို့ အင်္ဂလိပ်လိုဆိုရင် ရလဒ်ပိုကောင်းလေ့ရှိတဲ့အတွက် ဖြစ်ပါတယ်။

ရလာတဲ့ App ကို မြန်မာဘာသာနဲ့ လိုချင်ရင်လည်း ရပါတယ်။ အခုလို Prompt လေးပေးလိုက်ရုံပါပဲ။

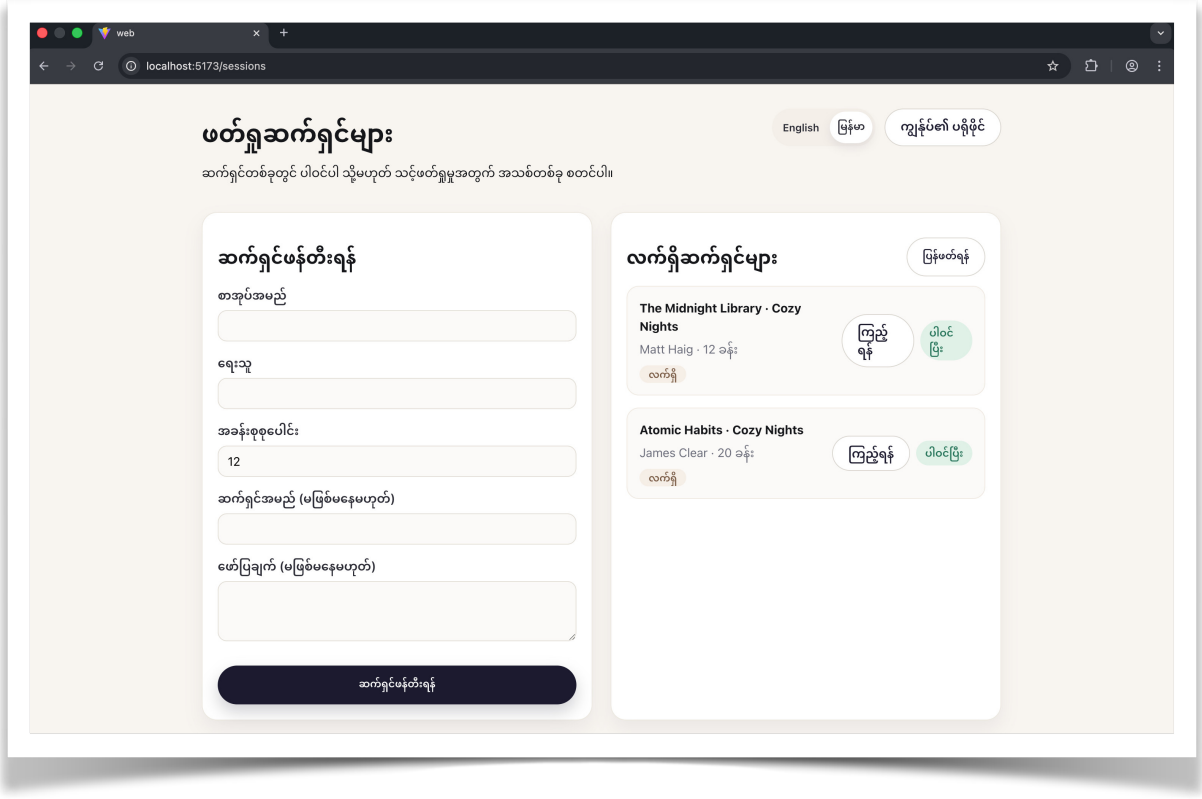
> *Review web app. Implement English and Burmese **dual language support**. Use language files, add translation strings for both language.*

အင်္ဂလိပ်၊ မြန်မာ နှစ်ဘာသာ Support လုပ်ပါ။ ဘာသာပြန်ဖိုင် တွေသုံးပါလို့ ပြောလိုက်တာပါ။

ဒါဟာ စကတည်းက မြန်မာလို UI နဲ့ လုပ်ခိုင်းလိုက်တာထက် ပိုကောင်းတဲ့နည်း ဖြစ်ပါတယ်။ ဘာဖြစ်လို့လဲဆိုတော့ မြန်မာလို UI မှာ AI ထည့်ပေးတဲ့ စာတွေက အကုန်အဆင်ပြေမှာ မဟုတ်ပါဘူး။ ကိုယ်က နောက်က လိုက်ပြင်ရမှာ သေချာပါတယ်။

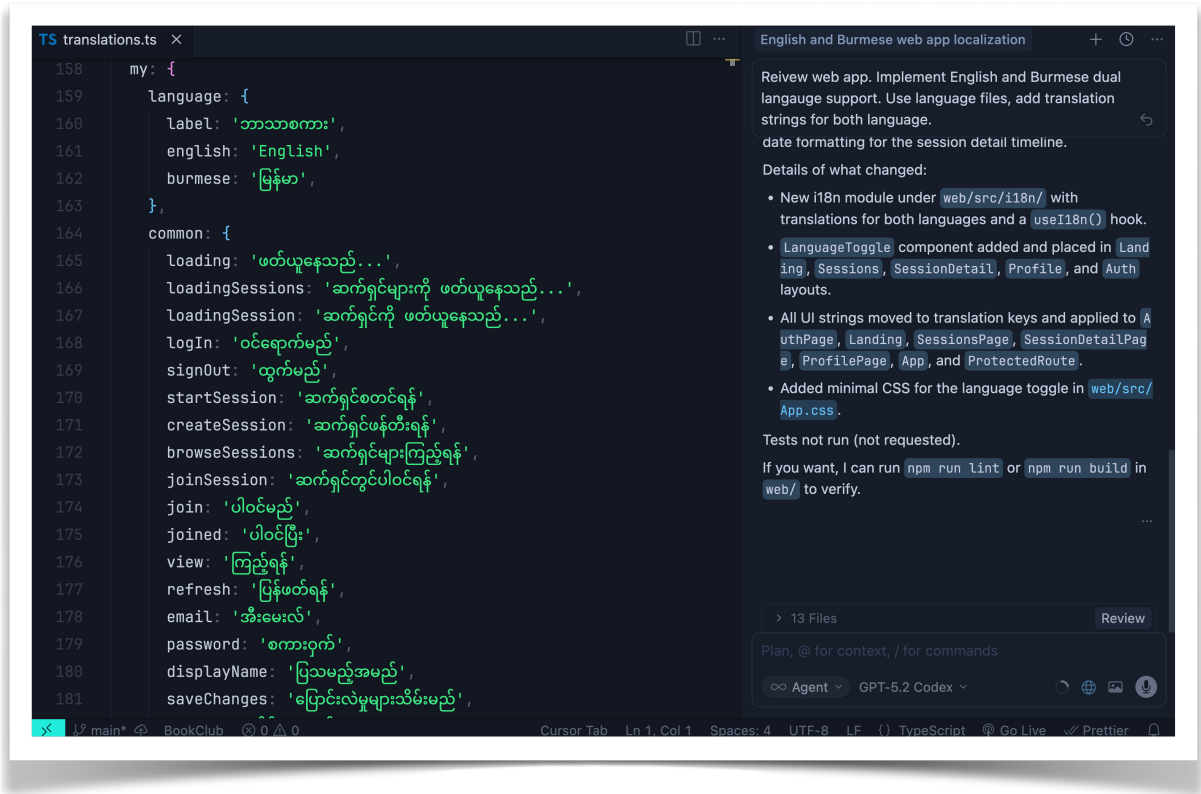
အဲ့ဒီလို ပြင်ဖို့လိုတဲ့အခါ ကုဒ်တွေထဲမှာ လိုက်ပြင်နေဖို့ မလိုတော့ဘဲ ဘာသာပြန် ဖိုင်ထဲမှာတင် အလွယ်တကူ ပြင်လို့ရသွားမှာပါ။

ဒီလုပ်ဆောင်ချက်လည်း တစ်ချက်တည်းနဲ့ ဘာမှထပ်ပြောစရာမလိုဘဲ အဆင်ပြေပြေ ရရှိသွားပါတယ်။



ပုံမှာတွေ့မြင်ရတဲ့အတိုင်း အများအားဖြင့် အဆင်ပြေပါတယ်။ ဒါပေမဲ့ ဟိုးထိပ်ဆုံးက “**ဖတ်ရှုဆက်ရှင်များ**” လို ဘာသာပြန်ချက်ကျတော့ သိပ်အဆင်မပြေပါဘူး။ တခြားပို သင့်တော်ရာ တခုခုနဲ့ ပြင်ဖို့လိုနိုင်ပါတယ်။ “**ကျွန်ုပ်၏ ပရိုဖိုင်**” ဆိုတာလည်း မဟုတ်သေးပါဘူး။ “**ပရိုဖိုင်**” လို့ပြင်လိုက်တာကမှ ပိုကောင်းနိုင်ပါသေးတယ်။ အဲ့ဒီလို ပြင်ဖို့လိုတာတွေရှိရင် စောစောက AI ထည့်ပေးသွားတဲ့ Translation ဖိုင်မှာ ဝင်ပြင်လိုက်ရုံပါပဲ။

ဒါကြောင့် အင်္ဂလိပ်၊ မြန်မာ နှစ်ဘာသာ Support မလုပ်ချင်ရင်တောင်မှ အင်္ဂလိပ်လိုပဲ အရင်လုပ်လိုက်ပြီး နောက်မှ အင်္ဂလိပ်၊ မြန်မာ နှစ်ဘာသာထည့်တာကို အားပေးပါတယ်။



အင်္ဂလိပ်လို မလိုအပ်ရင် မြန်မာကို Default ထားလိုက်ပြီး ဘာသာစကားပြောင်းတဲ့ ခလုတ်ကို ပြန်ဖြုတ်ထားလိုက်ရုံပါပဲ။

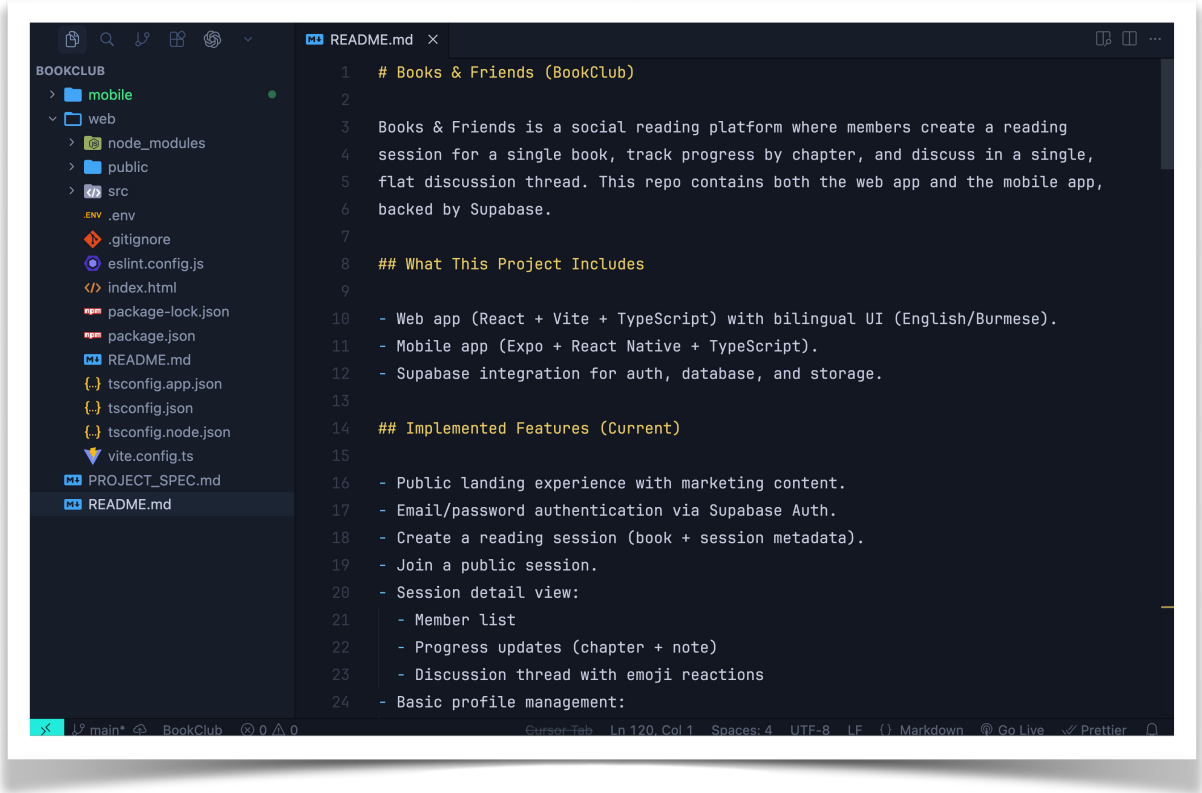
Documentation

ပရောဂျက် စစချင်း Spec လုပ်တုံးက နောက် Version မှာ ထပ်လုပ်မယ်လို့ ရည်ရွယ်ထားတဲ့ လုပ်ဆောင်ချက်တွေ ရှိခဲ့ပါတယ်။ ဒီနေရာမှာ ထည့်မဖော်ပြတော့ဘဲ စာရေးသူက ကိုယ့်ဘာသာ ဆက်လုပ်လိုက်တော့မှာပါ။

ဒီနေရာမှာတော့ နောက်ဆုံးတစ်ခုအနေနဲ့ ပရောဂျက် Documentation လေးထည့်လုပ်လိုက်ချင်ပါတယ်။ Agent Session သစ်တစ်ခု ဖွင့်လိုက်ပြီး အခုလို Prompt လေးပေးလိုက်ရုံပါပဲ။

> ***Read project spec**, analyze existing implementation, them **write a README file** to explain about both project and source code*

ဒါဆိုရင် ပရောဂျက်အကြောင်းကိုရော၊ ရေးထားတဲ့ ကုဒ်တွေအကြောင်းကိုပါ အတိုချုပ်ရှင်းပြထားတဲ့ Documentation လေးတစ်ခုကို အခုလို ရရှိသွားမှာပါ။



ဒါလေးက သိပ်မထူးဆန်းပေမဲ့ AI ကို ကုဒ်ရေးခိုင်းဖို့တင် မဟုတ်ဘဲ ပရောဂျက်ရဲ့ ကျန် အစိတ်အပိုင်းတွေမှာပါ အသုံးပြုသင့်တယ်ဆိုတာ သတိပြုမိစေချင်လို့ပါ။

ပရောဂျက်တစ်ခုမှာ ပါဝင်တဲ့ အစိတ်အပိုင်းတွေ အများကြီး ရှိနိုင်ပါတယ်။ အခု သုံးပိုင်း ကို လုပ်ခဲ့လိုက်တာပါ။

- Planning
- Writing Code
- Documentation

Debugging ခေါ် အမှားရှာခြင်း၊ Testing ခေါ် စမ်းသပ်ခြင်း၊ Distribution ခေါ် ဖြန့်ဝေ ခြင်းလို အဆင့်တွေမှာလည်း AI ကို အသုံးချလို့ ရပါသေးတယ်။

ဒီစာအုပ်က Software Development ကို In-Depth လေ့လာဖို့ ရည်ရွယ်တာ မဟုတ်လို့ Debugging တွေ Testing တွေအကြောင်းတော့ ထည့်မပြောနိုင်တော့ပါဘူး။ Distribution အကြောင်းကိုတော့ စာဖတ်သူ ကိုယ်တိုင်လိုက်လုပ်ကြည့်လို့ရအောင် **အခန်း (၈)** ရောက် တဲ့အခါ ထည့်သွင်းဖော်ပြပေးသွားမှာပါ။

ဒီအခန်းရဲ့ အဓိကရည်ရွယ်ချက်ကတော့ ဆန်းပြားတဲ့ နည်းစနစ်တယ်၊ ဆန်းကျယ်တဲ့ လျှို့ဝှက်ချက်တွေ တစ်ခုမှမလိုဘဲ၊ သိသင့်တဲ့ ဗဟုသုတတချို့ ရှိထားမယ်၊ System Thinking လေးကောင်းမယ်၊ ကြိုတင်ပြင်ဆင်မှုလေး ကောင်းမယ်ဆိုရင် တော်တော် ကျယ်ပြန့်တဲ့ ပရောဂျက်တွေ လုပ်လို့ရတယ် ဆိုတာကို လက်တွေ့နမူနာ ပြလိုက်တာပဲ ဖြစ် ပါတယ်။

ဒီနမူနာနဲ့ ဗဟုသုတတွေကို ကိုယ်တိုင် ဘယ်လောက်ထိ ဆက်လက် အသုံးချနိုင်လဲ ဆိုတာတော့ စာဖတ်သူပေါ်မှာ မူတည်ပါလိမ့်မယ်။ များများဆက်လုပ်ကြည့်ပါလို့ တိုက်တွန်းပါတယ်။ လုပ်ရင်းနဲ့ အချိန်အဆက် သဘောတရားလေးတွေ ပိုမိလာပါလိမ့်မယ်။ ဖြစ်တတ်တဲ့ ပြဿနာလေးတွေ ဖြေရှင်းနည်းလေးတွေကို ပိုမြင်လာပါလိမ့်မယ်။

အခန်း (၇) - Agent နည်းစနစ်များ

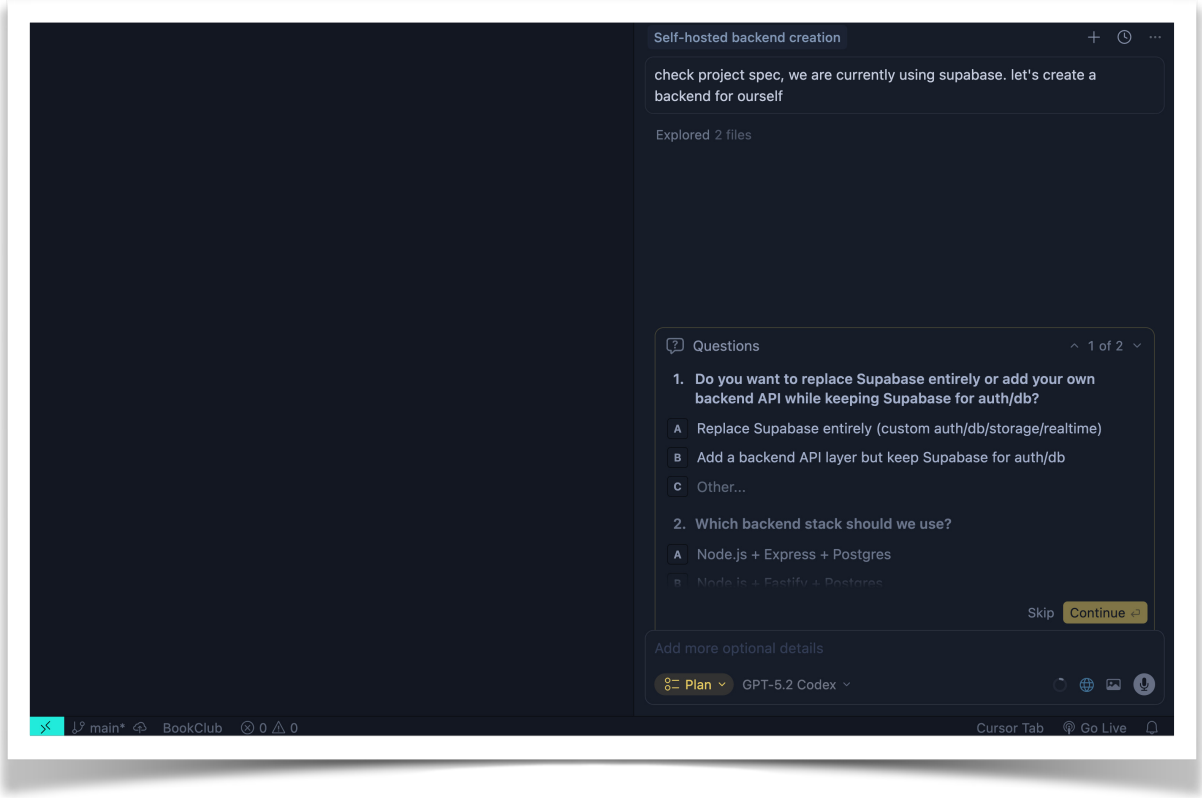
ဒီအခန်းမှာ Coding Agents တွေမှာ ပါဝင်လေ့ရှိတဲ့ ဖြည့်စွက်နည်းစနစ်တွေ အကြောင်း အကျဉ်းချုပ် ဖော်ပြချင်ပါတယ်။ အဲ့ဒါတွေကတော့ -

- Plan Mode
- Rules
- Skills
- Commands
- Subagents နဲ့
- MCP တို့ပဲဖြစ်ပါတယ်။

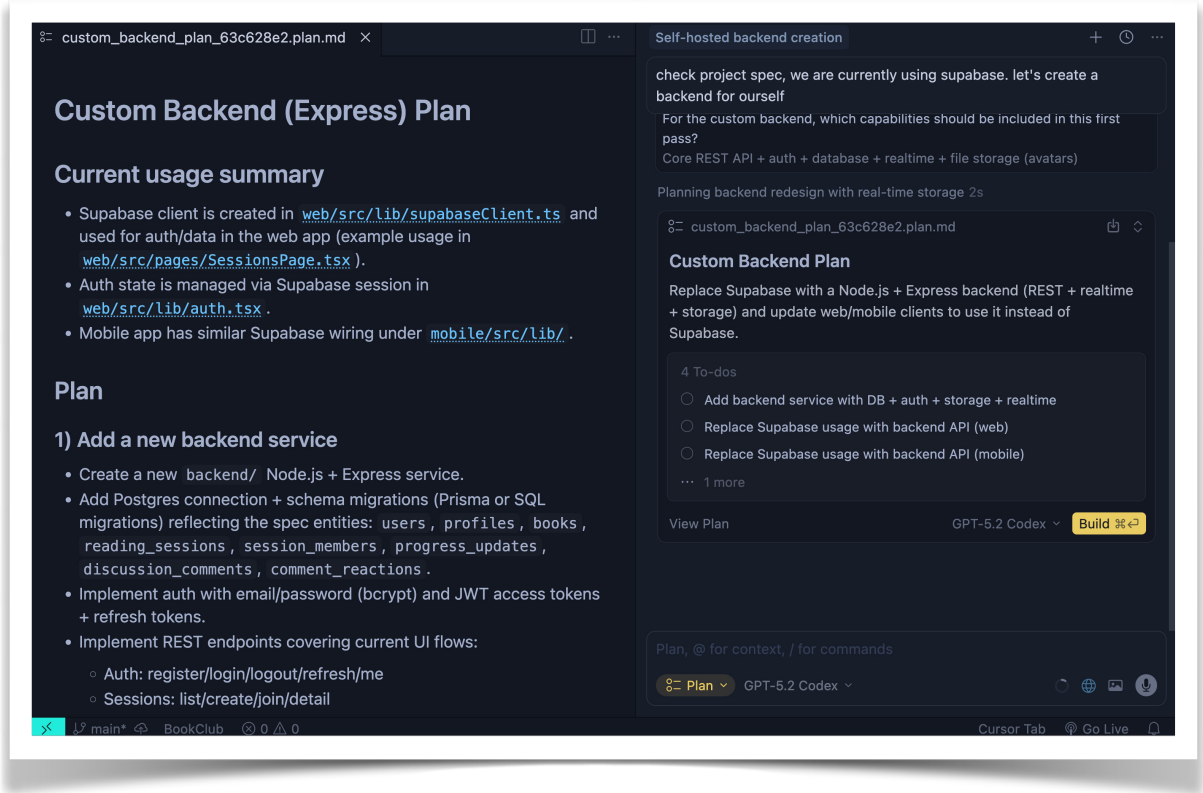
Plan Mode အကြောင်းကို ပြီးခဲ့တဲ့အခန်းမှာ နည်းနည်း ထည့်ပြောဖြစ်ခဲ့ပါတယ်။ Cursor ရဲ့ Plan Mode ကိုမသုံးဘဲ Agent နဲ့ Project Spec ဖန်တီးခဲ့ပါတယ်။ ကိုယ်က

တောင်းဆိုတဲ့အတွက် Agent က သူသိချင်တဲ့ မေးခွန်းတချို့ ကိုယ့်ကို ပြန်မေးတာကို လည်း တွေ့ခဲ့ကြရပါတယ်။

အကယ်၍ အလားတူလုပ်ဆောင်ချက်ကိုသာ Agent Mode နဲ့ မသုံးဘဲ Plan Mode ကို သာ သုံးခဲ့မယ်ဆိုရင် အခုလိုဖြစ်မှာပါ။



“**Backend လေးတစ်ခုလောက် လုပ်ကြရအောင်**” လို့ ပြောလိုက်တာနဲ့ မေးသင့်တဲ့ မေးခွန်းတွေ အလိုအလျောက် ပြန်မေးလာပါတယ်။ အဲဒီ မေးခွန်းတွေကို Prompt ပေးပြီး ဖြေနေစရာလည်း မလိုပါဘူး။ သူကြိုထည့်ပေးထားတဲ့ Option ခလုတ်နှိပ်ပြီး ရွေးသွားရုံ ပါပဲ။



အားလုံးဖြေပြီးသွားတဲ့အခါမှာတော့ အခုလို Detail Plan တစ်ခု ထွက်လာမှာပါ။ **Build** ခလုတ်ကို နှိပ်ပြီး ကုဒ်တွေ စရေးခိုင်းလို့ ရပါပြီ။

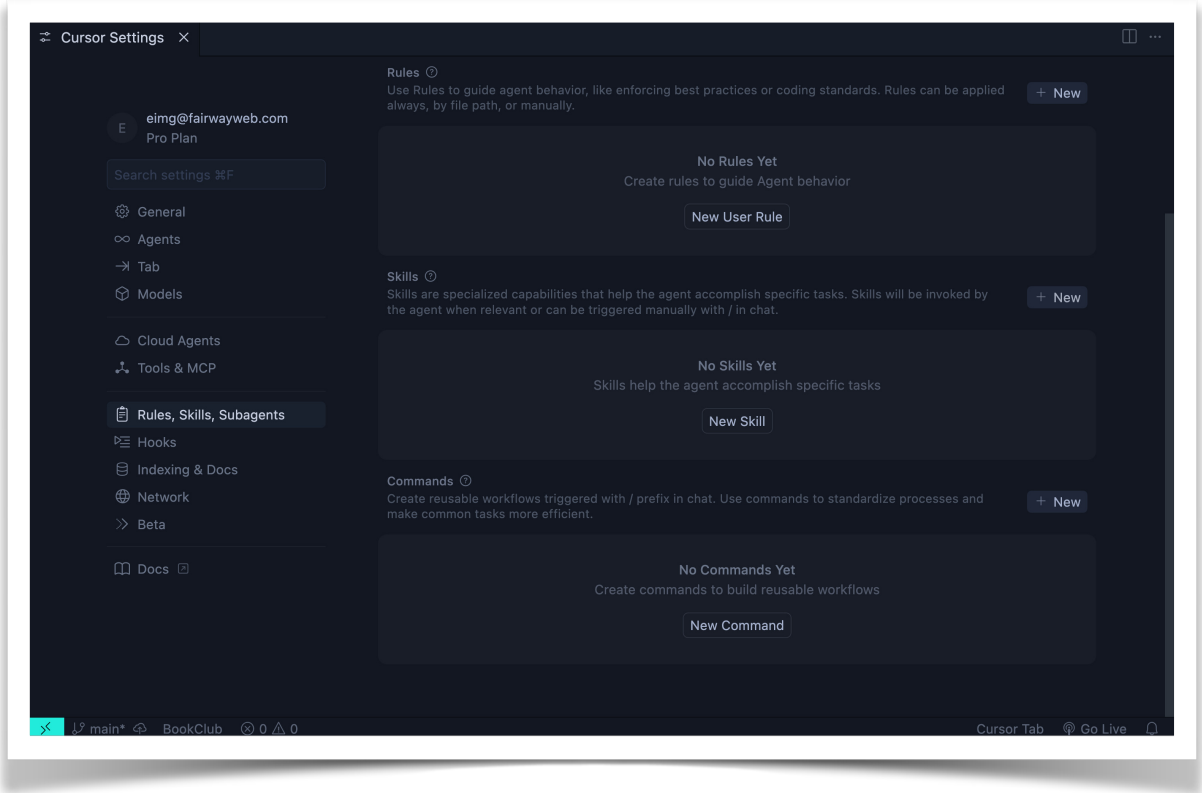
Plan Mode ဟာ စာရေးသူ အမြဲအသုံးပြုလေ့ရှိတဲ့ Feature ဖြစ်ပါတယ်။ ပရောဂျက် အစပိုင်းမှာသာမက၊ လုပ်ဆောင်ချက်အသစ်တစ်ခု ထည့်တော့မယ်ဆိုရင် တန်းမထည့်ခိုင်းဘဲ Plan Mode နဲ့ Detail Plan အရင်ထုတ်ပါတယ်။ ပြီးမှ ထည့်ခိုင်းပါတယ်။

Detail Plan ရသွားတာဟာ ရလဒ်ပိုကောင်းသွားတယ်ဆိုယုံထက် ပိုပါတယ်။ အဲ့ဒီ Detail Plan ကို ကြည့်ပြီး Build မလုပ်ခင် ထပ်ညှိချင်တာတွေပါ ညှိလို့ရသွားတာပါ။ ကိုယ်မစဉ်းစားမိတာတွေ ထည့်စဉ်းစားဖြစ်သွား စေပါတယ်။ တစ်ခါတစ်လေ စိတ်ကူးထဲက ပုံစံ

နဲ့ တကယ် Plan ချကြည့်လိုက်တော့ ထွက်လာတဲ့ပုံစံ သိပ်မကိုက်လို့ ပြန်ပြင်ရတာ ရှိပါတယ်။

<u>ကုဒ်တွေရေးပြီးမှ လိုက်ပြင်ရတာထက်စာရင် Plan အဆင့်မှာကတည်းက လိုအပ်တာတွေကို ကြိုပြင်လိုက်လို့ရသွားတာပါ။</u>

Cursor ရဲ့ **Settings** ကို သွားကြည့်လိုက်ရင် အခုလို လုပ်ဆောင်ချက်တွေကိုလည်း တွေ့ရနိုင်ပါတယ်။



Rules, Skills, Command, Subagents စတဲ့ လုပ်ဆောင်ချက်တွေပါ။

Rules ဆိုတာ ဘာတွေလုပ်သင့်တယ်၊ ဘာတွေမလုပ်သင့်ဘူး၊ ဘယ်လိုလုပ်သင့်တယ်၊ ဘယ်လိုမလုပ်သင့်ဘူး စသည်ဖြင့် Agent ကို ကိုယ့်ဘက်က ကန့်သတ်ပြောပြချင်တာတွေအတွက် သုံးရပါတယ်။

ဥပမာ - အမြဲတမ်း React နဲ့ Vite ကိုပဲ သုံးပါ။ သီးခြားတောင်းဆိုမှ Nextjs ကို သုံးပါ ဆိုတာမျိုးပါ။ ဒါဆိုရင် Agent က နောက်ပိုင်း ကိုယ်ထည့်ပြောနေစရာ မလိုဘဲ React နဲ့ Vite ကိုပဲ အဓိက သုံးပေးသွားမှာ ဖြစ်ပြီး၊ လိုအပ်လို့ ကိုယ့်ဘက်က Nextjs ကို သုံးပါလို့ ပြောမှသာ Nextjs ကို သုံးပေးတော့မှာပါ။

တချို့အသေးစိတ်တွေကတော့ Coding ပိုင်း သိတဲ့သူတွေအတွက်မှ ပိုအသုံးဝင်မှာပါ။
ဥပမာ - Database အတွက်ဆိုရင် Prisma 7 သုံးရမယ်။ React Router 6 ကို မသုံးရဘူး 7 ကို သုံးရမယ်။ SafeAreaView ကို မသုံးရဘူး safe-area-context ကို သုံးရမယ်။ စသည်ဖြင့် ကိုယ်သုံးစေချင်တဲ့ နည်းပညာတွေ၊ မသုံးစေချင်တဲ့ နည်းပညာတွေကအစ အကုန် ကြိုတင်သတ်မှတ်ပေးထားလို့ ရသွားတာပါ။

—

Skills ဆိုတာ ကြိုတင်သတ်မှတ်ထားတဲ့ “**လုပ်နည်းလုပ်ဟန်**” တွေပါ။

ဥပမာ - ကုဒ်ရဲ့အလုပ်လုပ်ပုံကို မေးလို့ ရှင်းပြတဲ့အခါ Diagram Flow လေးနဲ့ရှင်းပြရမယ်။ အရေးကြီးတဲ့ အပိုင်းလေးတွေ ဆွဲထုတ်ပြီး ရှင်းပြရမယ်။ ဘာကြောင့် ဒီလိုရေးတာလဲ၊ ဘယ်လိုဆိုရင် ပိုကောင်းမလဲ ရှင်းပြရမယ်လို့ Skill တစ်ခုအနေနဲ့ ကြိုတင်သတ်မှတ်ထားတယ် ဆိုကြပါစို့။

ကိုယ်က သိချင်လို့ “**ငါ့ကို ဒီကုဒ်တွေအကြောင်း ရှင်းပြကွာ**” လို့ ပြောလိုက်ရင် Agent က အဲ့ဒီ Skill ကို လှမ်းခေါ်ပြီး ကြိုတင်သတ်မှတ်ထားတဲ့ အတိုင်း Diagram Flow လေးတွေဆွဲပြီး အရေးကြီးကုဒ်လေးတွေ ရွေးထုတ် ရှင်းပြပေးသွားမှာပါ။

ကိုယ့်ဘာသာ Skills တွေ ဖန်တီးထားလို့ရသလို၊ သူများတွေ ဖန်တီးပေးထားတဲ့ Skills တွေထဲက ကိုယ့်အတွက် အသုံးဝင်လောက်တဲ့ Skills တွေရှာပြီး ယူထည့်ထားလို့လည်း ရပါတယ်။

—

Commands ဆိုတာကတော့ ထပ်ခါထပ်ခါ Agent ကို ပေးရတဲ့ Prompt တွေကို ကြိုရေးပြီး လိုတဲ့အခါ ပြန်ခေါ်သုံးလို့ရသွားတာပါ။

ဥပမာ - Section အသစ်ဖွင့်တိုင်း Agent ကို Spec ဖတ်ဖို့ ပြောရပါတယ်။ ရေးလက်စ ကုဒ်ကို Review လုပ်ဖို့ ပြောရပါတယ်။ အဲ့ဒီ Prompt ကို `analyze` လို့ခေါ်တဲ့ Command အဖြစ် သိမ်းထားလိုက်ရင် ထပ်ရိုက်စရာမလိုတော့ပါဘူး။ လိုအပ်တဲ့အခါ Prompt Box ထဲမှာ `/analyze` ဆိုပြီး လှမ်းခေါ်လိုက်ရုံပါပဲ။

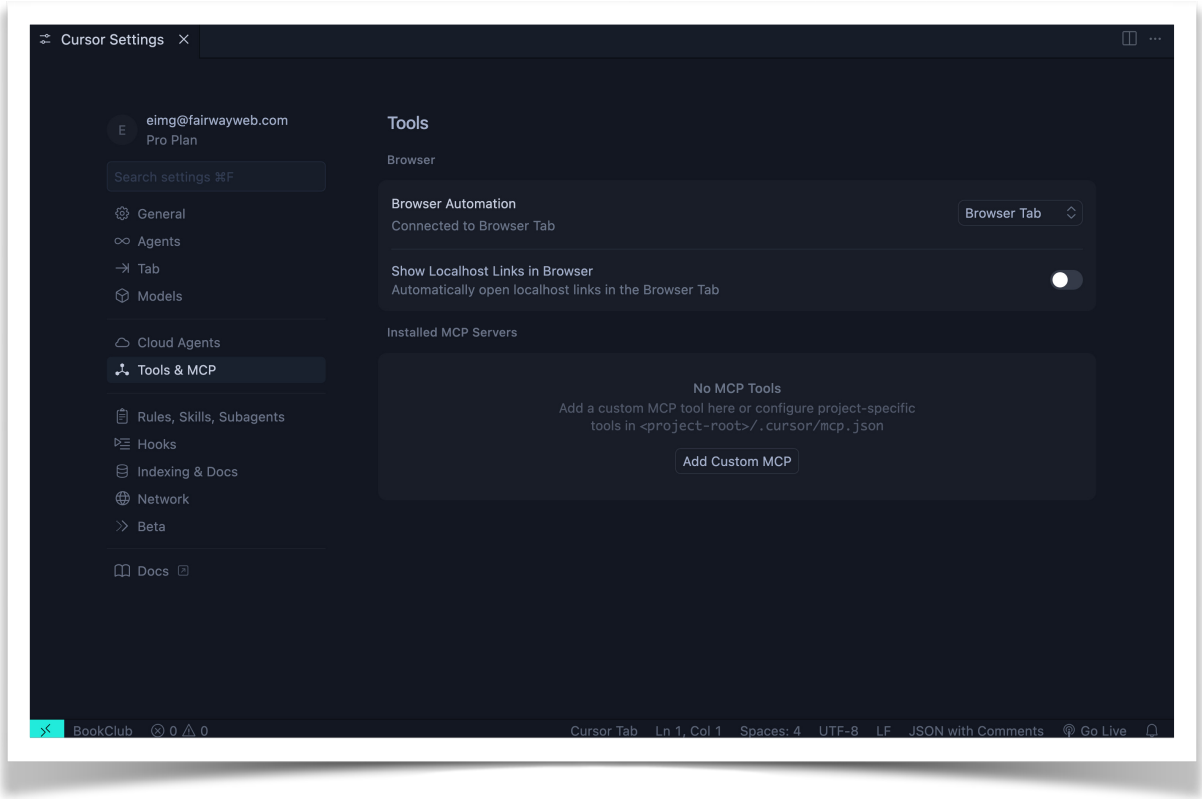
—

Subagent ဆိုတာတော့ ခေါ်သုံးလို့ရတဲ့ Agent Session တစ်ခုလို့ မြင်ကြည့်နိုင်ပါတယ်။

ဥပမာ - ကိုယ့်ဘာသာလည်း Prompt ပေးပြီး Code တွေကို Review လုပ်ခိုင်း၊ ဖြစ်နိုင်ခြေရှိတဲ့ ပြဿနာတွေ ရှာခိုင်းလို့ရပါတယ်။ Agent က ရှာပေးသွားမှာပါပဲ။ အဲ့ဒီ လုပ်ဆောင်ချက်ကို Subagent အနေနဲ့ သိမ်းထားလိုက်လို့ ရပါတယ်။

Subagent တွေသာ ကြိုရှိထားမယ်ဆိုရင် ရိုးရိုး Agent နဲ့ အလုပ်တစ်ခု ခိုင်းလိုက်တဲ့အခါ သူက Subagent တွေကို သင့်တော်သလို အလိုအလျောက် ခေါ်သုံးပေးသွားမှာပါ။ ဆိုလိုတာက ကုဒ်တွေရေးပြီးတာနဲ့ Review လုပ်ပေးတဲ့ Subagent, Test လုပ်ပေးတဲ့ Subagent, Documentation ကို Update လုပ်ပေးတဲ့ Subagent တွေကို အလိုအလျောက် ခေါ်သွားမယ်ဆိုရင် Agent လေးငါးခု စုပေါင်းပြီး အလုပ်လုပ်တဲ့ သဘော ဖြစ်သွားမှာပါ။

Cursor ရဲ့ **Settings** တွေထဲမှာ **MCP** ဆိုတာလည်း ရှိပါသေးတယ်။



MCP ဆိုတာ Model Context Protocol ရဲ့ အတိုကောက်ပါ။

Model က တချို့အချက်အလက်တွေ လိုချင်ရင် အင်တာနက်ကနေ ရှာနိုင်ပါတယ်။ အင်တာနက်ကနေ ရှာမယ့်အစား MCP ဆာဗာဆီကနေလည်း လှမ်းတောင်းလို့ ရတယ်လို့ အကျဉ်းချုပ် မှတ်နိုင်ပါတယ်။

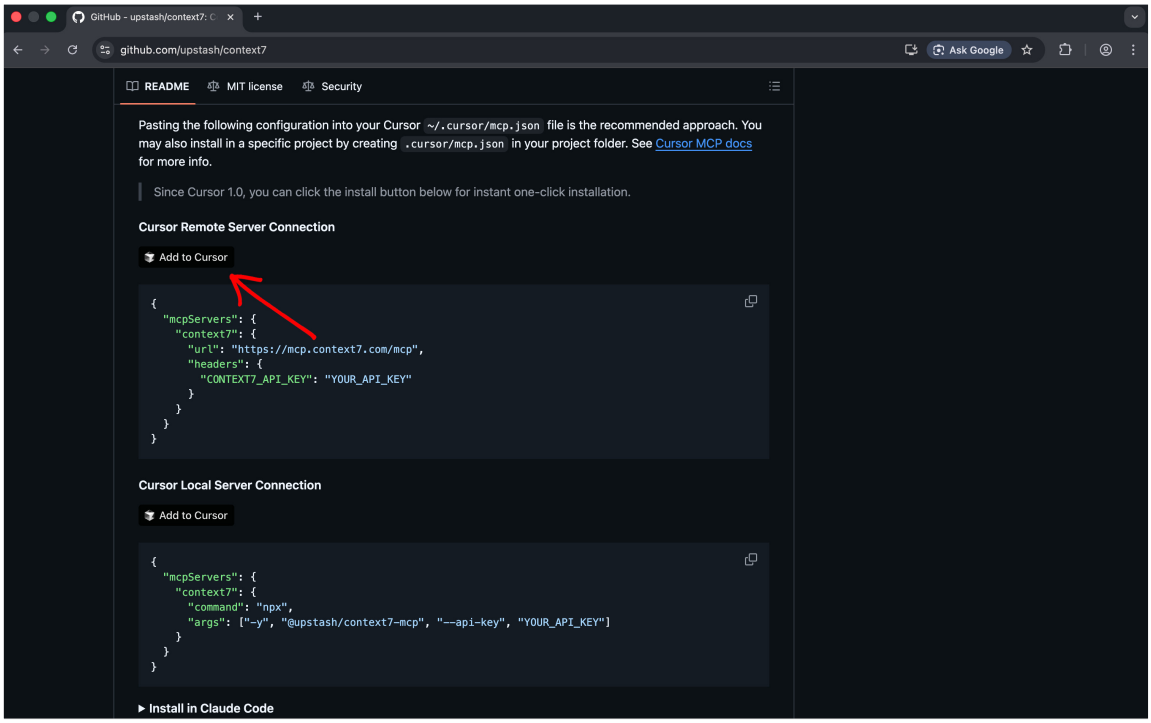
အင်တာနက်က ရတဲ့ အချက်အလက်တွေက လွဲနေတာ၊ ဒိတ်အောက်နေတာ ဖြစ်နိုင်သလို၊ တချို့အချက်အလက်တွေက အင်တာနက်မှာ Public အနေနဲ့ ရှိကို မရှိတာလည်း ဖြစ်နိုင်ပါတယ်။ MCP ဆာဗာတွေက Model လိုအပ်မဲ့ အချက်အလက်တွေကို ပိုပြီးတော့ တိတိကျကျ ပေးနိုင်ပါတယ်။

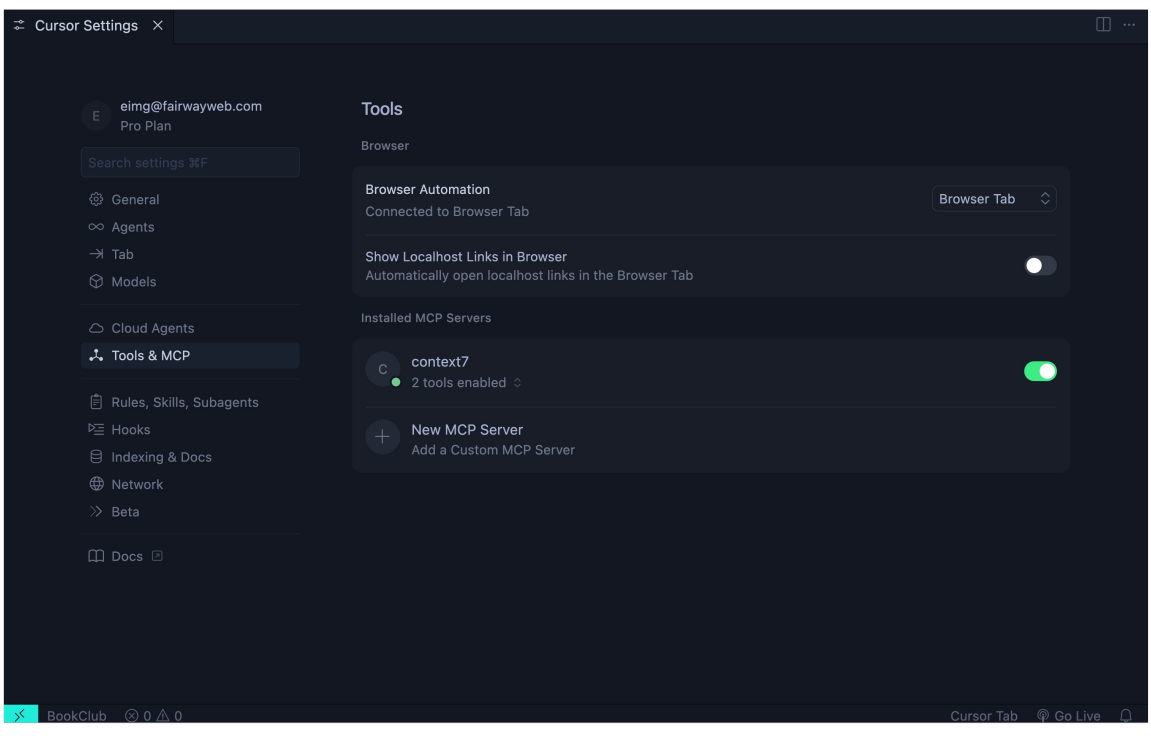
တော်တော်အရေးပါတဲ့ နည်းပညာတစ်ခုအဖြစ် ယူဆခဲ့ကြပေမဲ့ လက်တွေ့မှာ ဒီလောက်ကြီး ထိရောက်မှု မရှိသလိုဖြစ်နေတာလည်း ထူးခြားချက်တစ်ခုပါ။ နောက်ပိုင်း နည်းပညာအပြောင်းအလဲတွေနဲ့ ပြန်အသုံးဝင်ကောင်း ဝင်လာနိုင်ပါတယ်။

စမ်းကြည့်ချင်ရင်တော့ Context7 လို လက်တွေ့အသုံးဝင်တဲ့ MCP တစ်ခုနဲ့ စမ်းကြည့်လို့ ရပါတယ်။ ဒီလင့်ကို သွားလိုက်ပါ။

https://github.com/upstash/context7

Install in Cursor ကိုရှာပါ။ ပြီးတဲ့အခါ ပုံမှာမျှားပြထားတဲ့နေရာက **Add to Cursor** ကို နှိပ်လိုက်ရင် Context 7 MCP ကို ကိုယ့်စက်ထဲက Cursor မှာ ထည့်ပေးသွားမှာ ဖြစ်ပါတယ်။


GitHub - upstash/context7: C
github.com/upstash/context7
Ask Google
README
MIT license
Security
Pasting the following configuration into your Cursor ~/.cursor/mcp.json file is the recommended approach. You may also install in a specific project by creating .cursor/mcp.json in your project folder. See Cursor MCP docs for more info.
Since Cursor 1.0, you can click the install button below for instant one-click installation.
Cursor Remote Server Connection
Add to Cursor
{
"mcpServers": {
"context7": {
"url": "https://mcp.context7.com/mcp",
"headers": {
"CONTEXT7_API_KEY": "YOUR_API_KEY"
}
}
}
}
Cursor Local Server Connection
Add to Cursor
{
"mcpServers": {
"context7": {
"command": "npx",
"args": ["-y", "@upstash/context7-mcp", "--api-key", "YOUR_API_KEY"]
}
}
}
▶ Install in Claude Code


Cursor Settings
eimg@fairwayweb.com
Pro Plan
Search settings ⌘F
General
Agents
Tab
Models
Cloud Agents
Tools & MCP
Rules, Skills, Subagents
Hooks
Indexing & Docs
Network
Beta
Docs
Tools
Browser
Browser Automation
Connected to Browser Tab
Browser Tab
Show Localhost Links in Browser
Automatically open localhost links in the Browser Tab
Installed MCP Servers
context7
2 tools enabled
New MCP Server
Add a Custom MCP Server
BookClub
Cursor Tab
Go Live

Cursor ဝက်ဘ်ဆိုက်ကို အလိုအလျောက် ရောက်သွားပြီး Install ခလုတ်နှိပ်ပေးလိုက်ရင် ကိုယ့်စက်ထဲက Cursor ရဲ့ **MCP Setting** မှာ အခုလို **Context 7** ရှိနေတာကို တွေ့ရပါလိမ့်မယ်။

ဒီအတိုင်း စတင်အသုံးပြုလို့ရပါပြီ။ ပိုပြည့်စုံချင်ရင်တော့ Context 7 ဝက်ဘ်ဆိုက်ကိုသွား၊ အကောင့်ဆောက်၊ လော့ဂ်အင်ဝင်ပြီး API Key ယူရပါတယ်။

https://context7.com/

လက်ရှိ Install လုပ်ထားတဲ့ Context 7 MCP ကို Edit လုပ်ပြီး အဲ့ဒီ API Key ကို ထည့်ပေးရပါတယ်။

Context 7 ဆိုတာ Update Documentation ကို ပေးနိုင်တဲ့ နည်းပညာပါ။ ဒါကြောင့် Agent က တခုခုသိချင်တာ ရှာဖို့လိုရင် အင်တာနက်ပေါ်မှာ သွားရှာနေစရာမလိုတော့ဘဲ Context 7 ကို လှမ်းခေါ်ပြီး Documentation ရယူသွားမှာပါ။

ဒါကိုကြည့်လိုက်ရင် MCP ဆိုတာ လိုအပ်တဲ့အခါ Agent ကခေါ်သုံးလို့ရတဲ့ Api တမျိုး (သို့မဟုတ်) Service တမျိုးဖြစ်တယ် လို့ မြင်ကြည့်လို့ရပါတယ်။

စာရေးသူ ကိုယ်တိုင်ကတော့ Plan Mode တစ်ခုကိုပဲ အဓိက သုံးဖြစ်ပါတယ်။ Rules, Skills, Commands, Subagent, MCP စတာတွေ သိပ်မသုံးဖြစ်ပါဘူး။ သုံးမယ်ဆိုရင် သူ့နေရာနဲ့သူ အသုံးဝင်နိုင်ပေမဲ့ ရိုးရိုးရှင်းရှင်းပဲ အလုပ်လုပ်ရတာကြိုက်လို့ပါ။

ကျွမ်းကျင်သူများအကြားမှာကိုပဲ အုပ်စုနှစ်စု ကွဲနေပါတယ်။ Subagent ပေါင်းများစွာ၊ Skills ပေါင်းများစွာနဲ့ အလုပ်လုပ်နေသူတွေရှိကြသလို၊ Agent ရှိရင် ရပြီဆိုပြီး ဘာလိုလို Agent ကို ပြောလိုက်ရင်ရတာပဲ လို့ သဘောထားသူတွေလည်း ရှိကြပါတယ်။ သူ့အကြိုက်နဲ့ သူပါပဲ။

ကိုယ်တိုင်အားလုံး စုံအောင်မသုံးဖြစ်ပေမယ့် အရေးပါ အသုံးဝင်တဲ့ နည်းပညာတွေမို့လို့ အခုလို အကျဉ်းချုပ် ထည့်သွင်းဖော်ပြပေးလိုက်တာပါ။

အခန်း (၈) - Deployment - ဖြန့်ဝေခြင်း

ဒီအခန်းမှာ Vibe Coding လုပ် ရေးသားဖန်တီးထားတဲ့ App တွေကို အများအသုံးပြုလို့ ရအောင် ဘယ်လို Publish လုပ်ရမလဲဆိုတာ ဖော်ပြသွားမှာပါ။ ဒီအလုပ်ကို Deployment လုပ်တယ် လို့လည်း ခေါ်နိုင်ပါတယ်။

Coding လိုပါပဲ၊ Deployment ကဏ္ဍဟာလည်း ဆာဗာတွေ၊ နက်ဝပ်တွေ၊ ကွန်တိန်နာတွေ နဲ့ အလွန်ကျယ်ပြန့်တဲ့ ဘာသာရပ်ဖြစ်ပါတယ်။ နည်းလမ်းအမျိုးမျိုး ရှိတဲ့ထဲက Coding တွေ Deployment တွေ မသိတဲ့သူများနဲ့ သင့်တော်နိုင်တဲ့ နည်းလမ်းကို အရင်ဖော်ပြပေးပါမယ်။

ဒီနည်းလမ်းကို အသုံးချနိုင်ဖို့ ပထမဆုံး စတင်လေ့လာရမှာက Git နဲ့ GitHub လို့ခေါ်တဲ့ နည်းပညာနှစ်ခုပါ။

Git & GitHub

Git ဆိုတာ ရေးထားတဲ့ကုဒ်တွေကို စီမံလို့ရတဲ့ နည်းပညာပါ။ စီမံတယ်ဆိုတာ ကုဒ်တွေကို မှတ်တမ်းအလိုက်သိမ်း၊ သိမ်းထားတဲ့ မှတ်တမ်းတွေကို လိုအပ်ရင် ပြန်သွား၊ ဒါမျိုးတွေ လုပ်လို့ရတဲ့ နည်းပညာပါ။

သူကိုယ်တိုင်ကလည်း ကျယ်ပြန့်တဲ့ နည်းပညာတစ်ခုပါပဲ။ လိုတိုရှင်း အကျဉ်းချုပ်လေးပြောပြပါမယ်။ ပထမဆုံး အောက်ပါလိပ်စာကနေ Download လုပ်ပြီး Git ကို Install လုပ်နိုင်ပါတယ်။

https://git-scm.com/

Install လုပ်ပြီးပြီဆိုရင် ကုဒ်တွေကို စတင်စီမံလို့ရပါပြီ။

ကွန်ပျူတာမရှိလို့ ဒါတွေလုပ်လို့မရဘူး (သို့မဟုတ်) မလုပ်ချင်ဘူးဆိုရင် ကျော်လိုက်လို့ရပါတယ်။ ပိုပြည့်စုံအောင်သာ ထည့်ပြောပြတာပါ။ ပိုလွယ်တဲ့နည်းကို နောက်တစ်ဆင့်မှာ ထည့်ပြောပြပါမယ်။

ပရောဂျက်ဖိုဒါထဲမှာ Terminal ဖွင့်လိုက်ပါ။ Cursor ရဲ့ Terminal ကိုသုံးရင်လည်း ရပါတယ်။

```
git init
```

ဒီ Command ကို Run လိုက်ရင် လက်ရှိကုဒ်ဖိုဒါကို Git နဲ့ စတင်စီမံလိုက်တာပါပဲ။

ပြီးတဲ့အခါ အခုလို ထပ် Run လိုက်ပါ။

```
git add .
```

နောက်ဆုံးက Dot လေးပါတာကို သတိပြုပါ။ ဒီ Command က လက်ရှိ ပရောဂျက်ဖိုဒါထဲက ရှိသမျှဖိုင်အားလုံးကို Git ရဲ့ မှတ်တမ်းထဲမှာ သိမ်းဖို့ ထည့်လိုက်တာပါ။ မသိမ်းရသေးပါဘူး။ သိမ်းဖို့ထည့်လိုက်တဲ့အဆင့်ပါ။

```
git commit -m “your comment”
```

ဒီ Command က ကုဒ်တွေကို မှတ်တမ်းတစ်ခုအနေနဲ့ သိမ်းလိုက်တာပါ။ “**your comment**” နေရာမှာ ပေးချင်တဲ့ မှတ်ချက်တစ်ခု ပေးလို့ရပါတယ်။ ဘယ်လိုကုဒ်တွေပါဝင်တယ်၊ ဘာအတွက် ရေးထားတယ်ဆိုတာမျိုး မှတ်ချက်တွေကိုပေးပြီး နောင်ပြန်ကြည့်တဲ့အခါ သိရအောင် သိမ်းရတဲ့သဘောပါ။

ဒါဆိုရင် ပရောဂျက်ဖိုဒါထဲမှာ ရှိသမျှ ဖိုင်တွေအားလုံးကို မှတ်တမ်းတစ်ခုအဖြစ် သိမ်းပြီးသွားပါပြီ။ နောက်ပိုင်းမှာ ပြင်ဆင်မှုတွေ ထပ်လုပ်ပြီးတိုင်း၊ ဖြည့်စွက်မှုတွေ ထပ်လုပ်ပြီးတိုင်း အခုလို မှတ်တမ်းတွေ သိမ်းသွားနိုင်ပါတယ်။

```
git add .
git commit -m “your comment”
```

သိမ်းဖို့အရင်ထည့်။ ပြီးရင် သိမ်း ဆိုတဲ့ အဆင့်တွေအတိုင်း သွားရတာပါ။ ထုံးစံအတိုင်း ဒါတွေကို AI ကို လုပ်ခိုင်းရင်လည်း လုပ်ပေးပါတယ်။ Run ရမဲ့ Command တွေ မေ့သွားရင်လည်း ပြန်မေးကြည့်လို့ရပါတယ်။

```
git log
```

ဒါက သိမ်းထားတဲ့ မှတ်တမ်းတွေကို ပြန်ခေါ်ကြည့်တာပါ။ သိမ်းထားတဲ့မှတ်တမ်းတွေကို ပြန်သွားလို့ရပေမဲ့ ပြန်သွားနည်းတွေထိ ထည့်မပြောတော့ပါဘူး။ သွားချင်ရင် သွားလို့ရတယ်ဆိုတာလောက်ပဲ မှတ်ထားလိုက်ပါ။

နောက်တစ်ဆင့်အနေနဲ့ **GitHub** ကို ဆက်ကြည့်ကြပါမယ်။ GitHub ဆိုတာ ရေးထားတဲ့ကုဒ်တွေကို တင်ထားလို့ရတဲ့ နည်းပညာပါ။ အခမဲ့သုံးလို့ရပါတယ်။ ပထမဆုံး အောက်ပါလိပ်စာကိုသွား၊ အကောင့်ဆောက်ပြီး Login ဝင်ထားလိုက်ပါ။

https://github.com/

ပြီးတဲ့အခါ **GitHub CLI** လို့ခေါ်တဲ့ နည်းပညာကို Install လုပ်ရပါမယ်။ အောက်ပါလိပ်စာကိုသွားပြီး သူလမ်းညွှန်ထားတဲ့ နည်းအတိုင်း Install လုပ်လိုက်ပါ။

https://cli.github.com/

သူလမ်းညွှန်ထားတဲ့အတိုင်းလုပ်ပါလို့ ပြောလိုက်တာက အသုံးပြုထားတဲ့ Operating System ပေါ်မူတည်ပြီး Install လုပ်နည်း ကွဲပြားမှာ မို့လို့ပါ။

ပြီးသွားရင် Browser မှာ GitHub ဝက်ဘ်ဆိုက်အတွက် Login ဝင်ထားပြီး၊ Terminal မှာ အခုလိုရိုက်ထည့်ပါ။

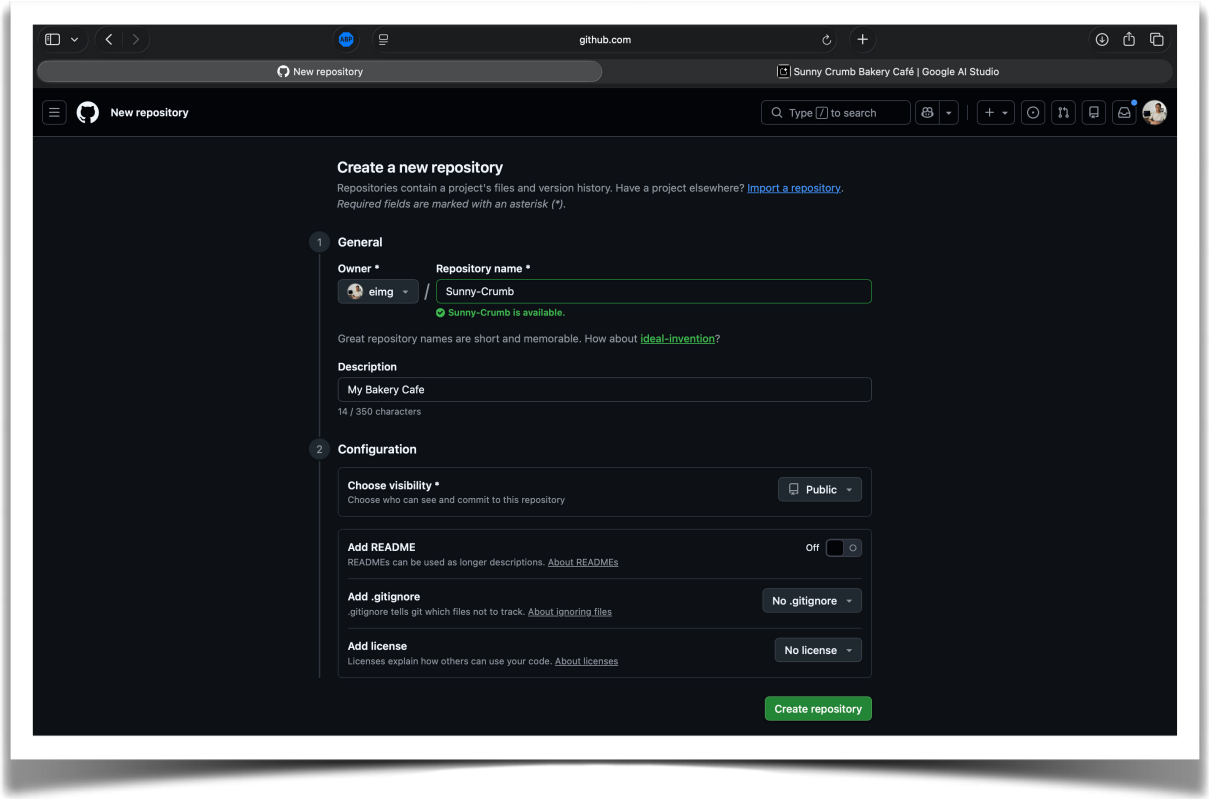
```
gh auth login
```

သူမေးတဲ့မေးခွန်းတွေ ဖြေလိုက်ပါ။ ဒါဆိုရင် ကိုယ့်စက်ရဲ့ Terminal မှာလည်း GitHub ကို Login ဝင်ထားပြီး ဖြစ်သွားပါပြီ။ အားလုံးစုံသွားပါပြီ။

Git ကို အသုံးပြုစီမံနေတဲ့ ကုဒ်တွေကို GitHub မှာ တင်ဖို့အတွက် ပထမဆုံး ပုံမှာမျှားပြထားတဲ့ ခလုတ်နှစ်ခုထဲကတစ်ခုကို နှိပ်ပြီး Repository အသစ်ဖန်တီးလိုက်ပါ။

ပြီးတဲ့အခါ နှစ်သက်ရာအမည်ပေးလိုက်ပါ။

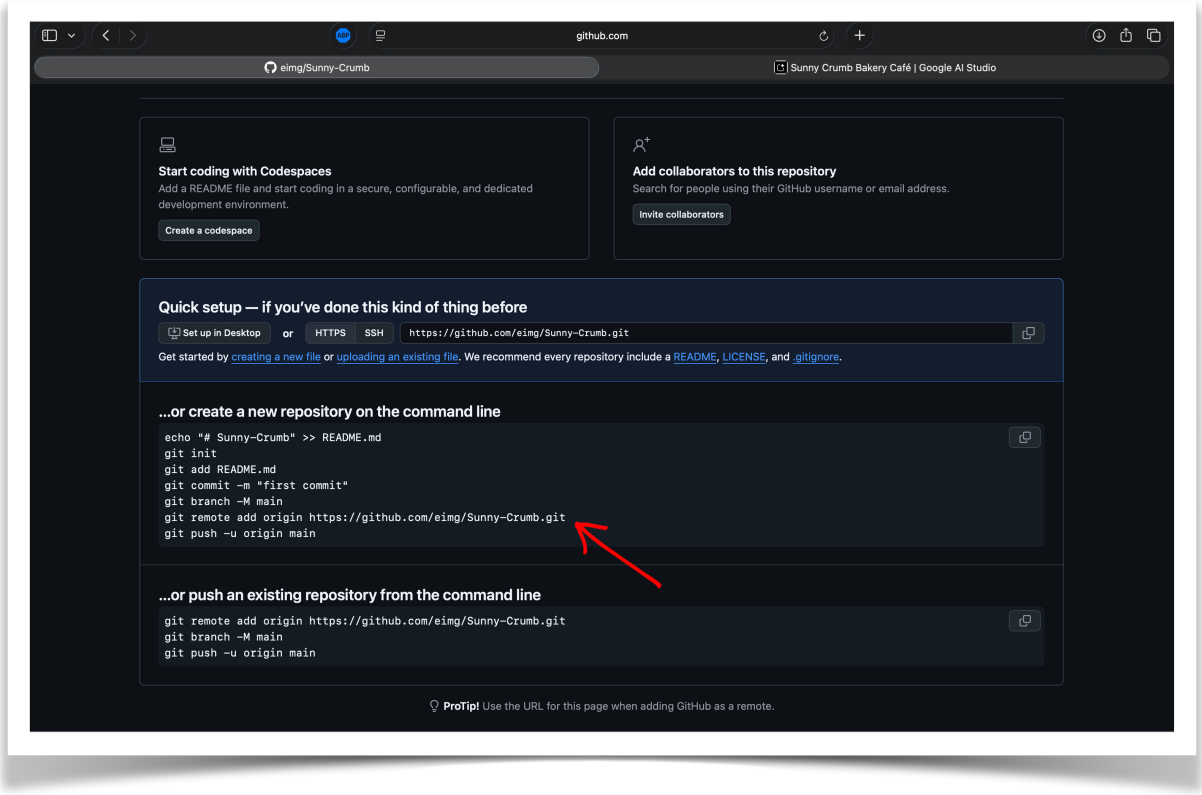
Public လား Private လား ရွေးလို့ရတာ သတိပြုပါ။ Public ဆိုရင် ကုဒ်တွေကို လူတိုင်းက ကြည့်လို့ရပြီး Private ဆိုရင်တော့ ကိုယ်ပဲကြည့်လို့ရမှာပါ။ ကျန်တဲ့ Option တွေတော့ ပြောင်းစရာမလိုပါဘူး။ ဒါပေမဲ့ သိသင့်ပါတယ်။



ကိုယ့်ရဲ့ ကုဒ်ဖိုဒါထဲမှာ README.md အမည်နဲ့ ဖိုင်တစ်ခုရှိမယ်ဆိုရင် ပြန်ကြည့်တဲ့အခါ အဲ့ဒီဖိုင်ကို အဓိက Content အနေနဲ့ ပြပေးမှာဖြစ်ပါတယ်။ ဒါကြောင့် ပြီးခဲ့တဲ့အခန်းမှာ တုံးက ပရောဂျက် Documentation ကို **README.md** အမည်နဲ့ ရေးခိုင်းခဲ့တာပါ။

.gitignore ဖိုင်ကလည်း အရေးကြီးပါတယ်။ ရှေ့ဆုံးက Dot နဲ့ စပါတယ်။ အဲ့ဒါမပါလို့ မရပါဘူး။ မှတ်တမ်းထဲမှာ ထည့်မသိမ်းစေချင်တဲ့ ဖိုင်တွေရှိရင် `.gitignore` ထဲမှာ ရေးထည့်ထားလို့ ရပါတယ်။ အရေးကြီးတဲ့ Secret Key တွေ ပါလေ့ရှိတဲ့ `.env` ဖိုင်လိုဖိုင်တွေ၊ Download လုပ်ထားတဲ့ Package တွေရှိတဲ့ `node_modules` လို ဖိုဒါတွေဆိုရင် ထည့်သိမ်းလေ့ မရှိပါဘူး။ AI ကို `.gitignore` ဖိုင်လေး ရေးထည့်ပေးပါလို့ခိုင်းလိုက်ရင် အကောင်းဆုံးပါပဲ။

`README` တို့ `.gitignore` တို့ကို ကိုယ်စက်ထဲမှာပဲ လုပ်လိုက်ပါ။ New Repository ဖောင်မှာ ထည့်ရွေးမနေပါနဲ့။ New Repository ဖောင်မှာ အမည်လောက်ပဲပေးပြီး **Create Repository** ကို နှိပ်လိုက်ရင် ဘာဆက်လုပ်ရမလဲဆိုတဲ့ Instruction ကျလာပါလိမ့်မယ်။



ပုံမှာမျှားပြထားတဲ့လိုင်းက အရေးအကြီးဆုံးပါပဲ။ အဲ့ဒါ ကိုယ်စက်ထဲက Git ဖိုဒါနဲ့ GitHub ပေါ်မှာ အခုဖန်တီးတဲ့ Repository ကို ချိတ်ပေးလိုက်တာပါ။

သူ့အပေါ်က `git init` တို့ `git add` တို့ `git commit` တို့က လုပ်ပြီးသားဆိုရင် ထပ်လုပ်စရာမလိုပါဘူး။ ဒီ Command နှစ်ခုကို ကူးယူပြီး Run လိုက်ပါ။ ဒါဆိုရင် ကိုယ့်စက်ထဲက ကုဒ်တွေကို GitHub ပေါ်တင်ပြီး ဖြစ်သွားပါပြီ။

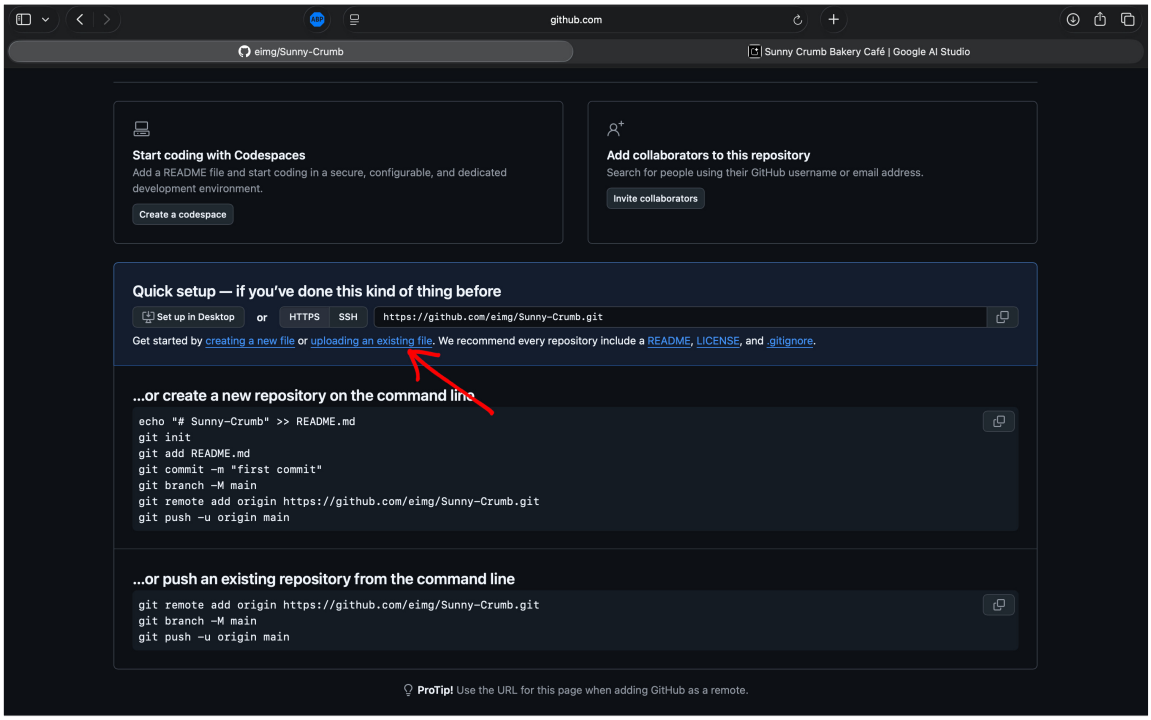
```
git remote add origin [repo url]
git push -u origin main
```

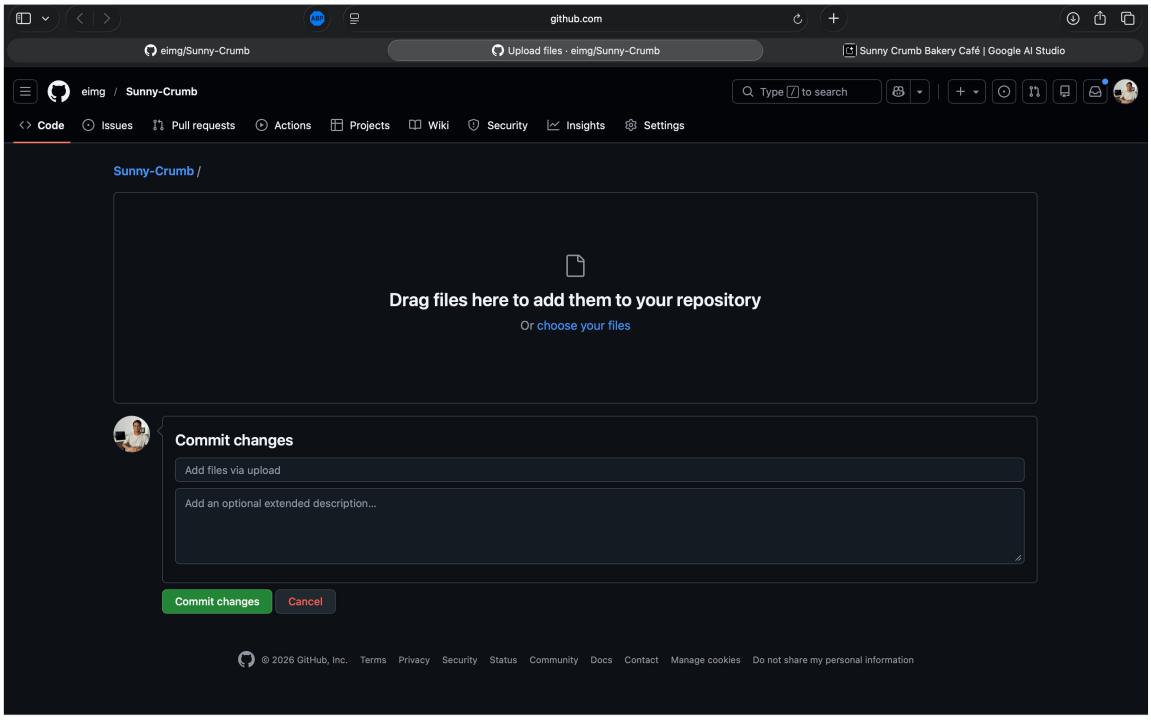
`[repo url]` နေရာမှာတော့ GitHub ကပေးတဲ့ URL အမှန်ဖြစ်ရပါမယ်။

File Upload (ပိုလွယ်တဲ့နည်း)

အဲ့ဒါတွေမလုပ်ချင်လို့ ပိုလွယ်တဲ့နည်းကို သိချင်ရင်တော့ နောက်စာမျက်နှာက ပုံမှာမျှား ပြထားတဲ့ လင့်ကို နှိပ်လိုက်ရုံပါပဲ။ ဘာ Command မှာ Run စရာမလိုပါဘူး။ ကွန်ပျူတာ လည်း မလိုပါဘူး။ Git လည်းမလိုပါဘူး။ GitHub အကောင့်ရှိရုံနဲ့ ရနိုင်ပါတယ်။

နောက်တစ်ဆင့်မှာ ကိုယ့်ပေးပို့လိုတဲ့ ကုဒ်ဖိုင်တွေကို ရွေး၊ မှတ်ချက် Commit changes ခလုတ်ကိုနှိပ်ပြီး သိမ်းလိုက်ရုံပါပဲ။ ဒါဆိုရင် ရှေ့ပိုင်းမှာ ပြောခဲ့တဲ့ Command တွေ Run နေစရာ မလိုဘဲ ကုဒ်တွေကို GitHub ပေါ်တင်လိုက်တာပါ။


github.com
eimg/Sunny-Crumb
Sunny Crumb Bakery Café | Google AI Studio
Start coding with Codespaces
Add a README file and start coding in a secure, configurable, and dedicated development environment.
Create a codespace
Add collaborators to this repository
Search for people using their GitHub username or email address.
Invite collaborators
Quick setup — if you've done this kind of thing before
Set up in Desktop
or
HTTPS
SSH
https://github.com/eimg/Sunny-Crumb.git
Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.
...or create a new repository on the command line
echo "# Sunny-Crumb" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/eimg/Sunny-Crumb.git
git push -u origin main
...or push an existing repository from the command line
git remote add origin https://github.com/eimg/Sunny-Crumb.git
git branch -M main
git push -u origin main
ProTip! Use the URL for this page when adding GitHub as a remote.


github.com
eimg/Sunny-Crumb
Upload files · eimg/Sunny-Crumb
Sunny Crumb Bakery Café | Google AI Studio
eimg / Sunny-Crumb
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Sunny-Crumb /
Drag files here to add them to your repository
Or choose your files
Commit changes
Add files via upload
Add an optional extended description...
Commit changes
Cancel
© 2026 GitHub, Inc.
Terms
Privacy
Security
Status
Community
Docs
Contact
Manage cookies
Do not share my personal information

နောက်တစ်ဆင့် ဆက်စမ်းကြည့်လို့ရအောင်၊ ရှေ့ပိုင်းမှာ AI Studio နဲ့လုပ်ထားလက်စရှိတဲ့ ကုဒ်တွေကို Download လုပ်ထားလိုက်ပါ။ ပြီးတဲ့အခါ GitHub ပေါ်တင်ထားလိုက်ပါ။

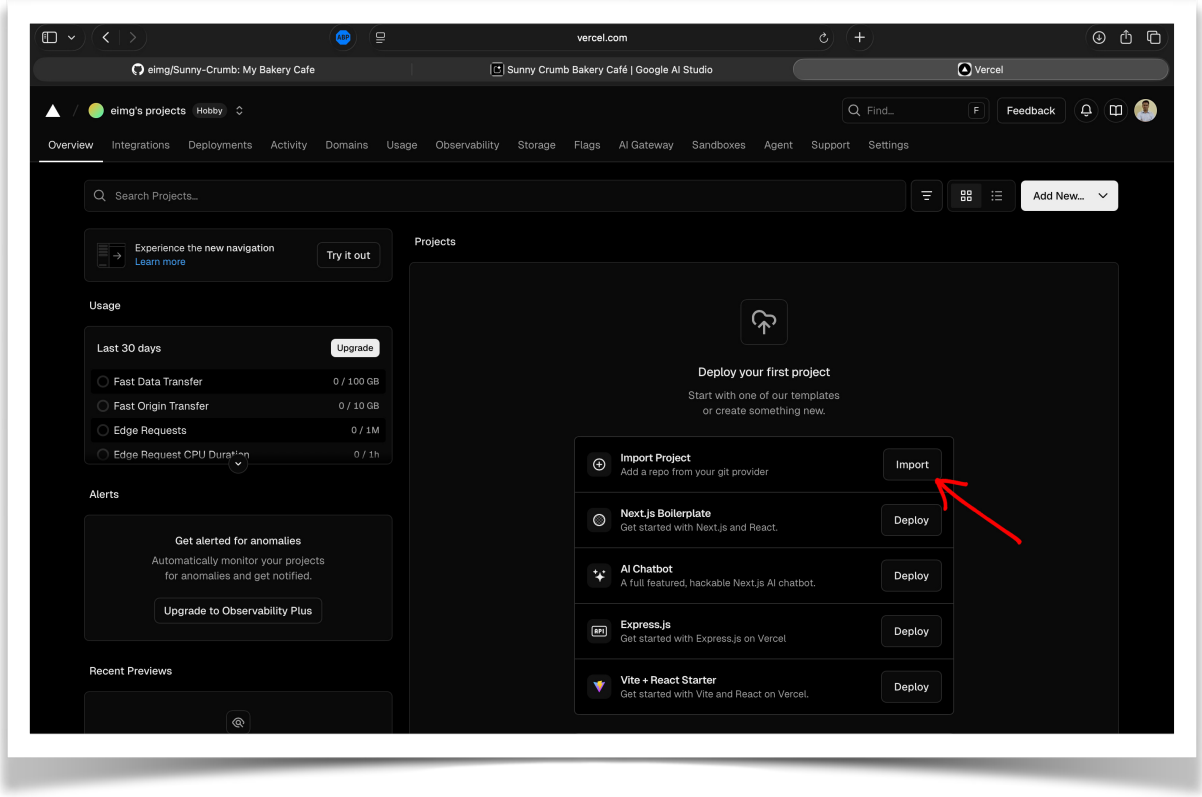
Publishing

လက်ရှိအချိန်မှာ ရေးထားတဲ့ကုဒ်တွေကို အများအသုံးပြုလို့ရအောင် Publish လုပ်ဖို့ အတွက် အလွယ်ဆုံးနဲ့ အမြန်ဆုံး ပလက်ဖောင်းကတော့ **Vercel** ဖြစ်ပါတယ်။ အခမဲ့ အသုံးပြုလို့ရပါတယ်။

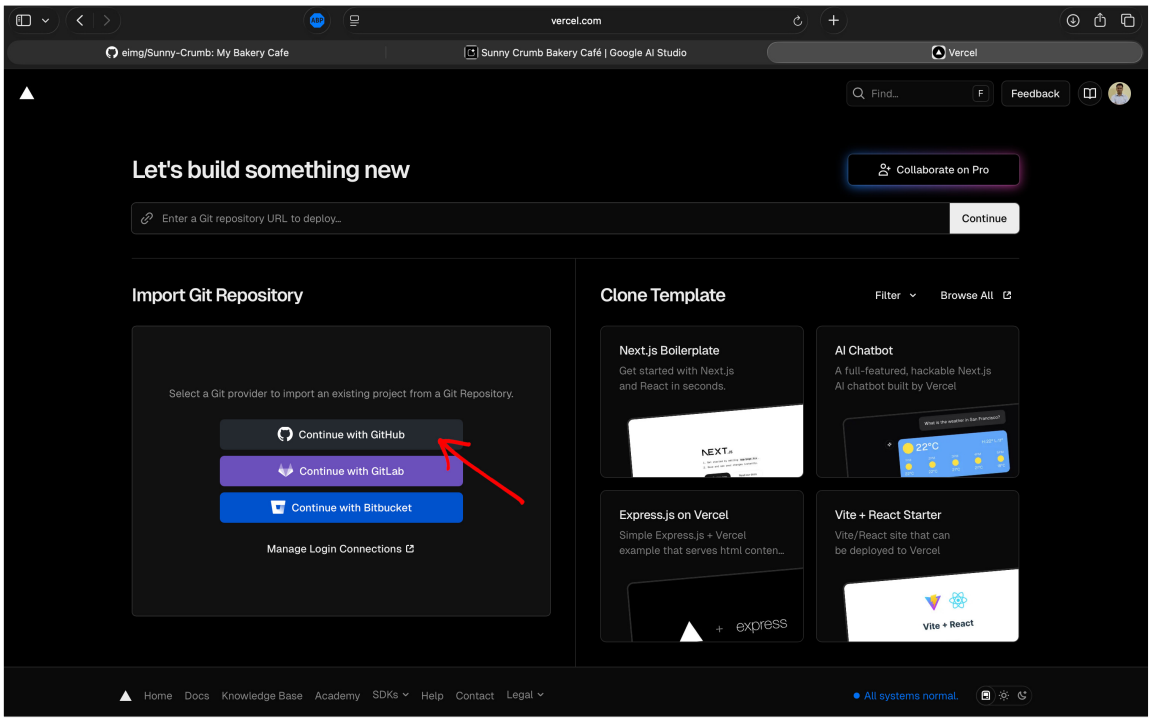
https://vercel.com

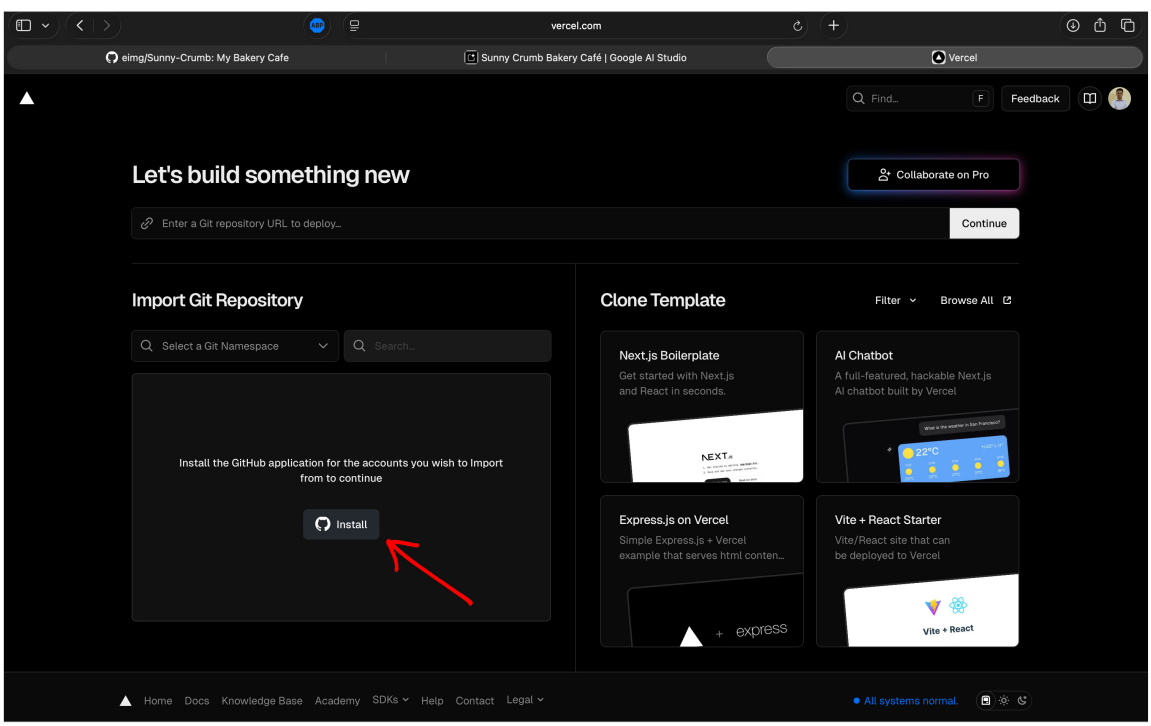
ပေးထားတဲ့လင့်ကိုသွားပြီး အကောင့်ဆောက်လိုက်ပါ။ အကောင့်သစ် မဆောက်ချင်ရင်လည်း စောစောက ဆောက်ထားတဲ့ GitHub အကောင့်နဲ့ Login ဝင်လို့ရပါတယ်။ Google အကောင့်တွေ ဘာတွေနဲ့လည်း ဝင်လို့ရပါတယ်။ တစ်ခုလို တစ်ခါအကောင့်သစ် ဆောက်မဲ့အစား GitHub အကောင့်တွေ Google အကောင့်တွေနဲ့ ဝင်လို့ရရင် ဝင်လိုက်တာက ပိုကောင်းပါတယ်။

Vercel မှာ Login ဝင်ပြီးရင် ပုံမှာမျှားပြထားတဲ့ Import ခလုတ်ကို နှိပ်လိုက်ပါ။



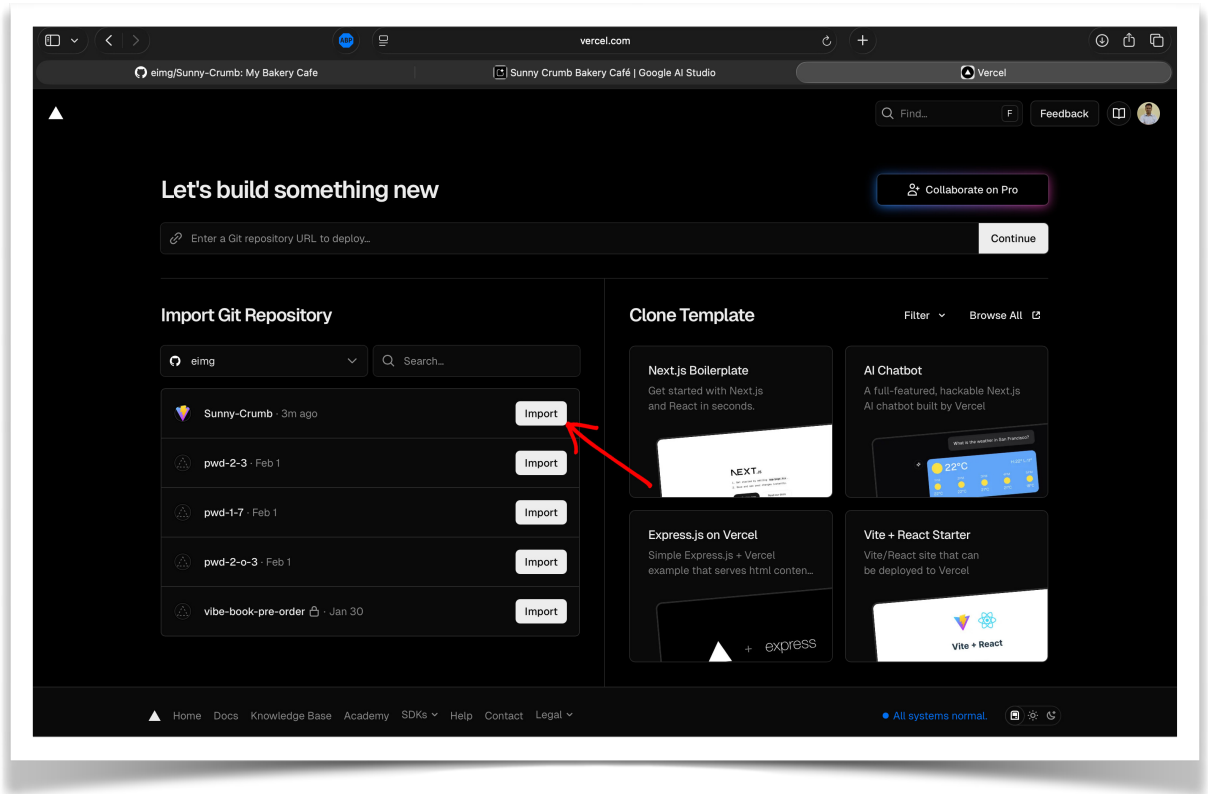
နောက်တစ်ဆင့်မှာ Continue with GitHub ကို နှိပ်လိုက်ပါ။ နောက်တစ်ဆင့်ကို ရောက်တဲ့အခါ Install ကို နှိပ်လိုက်ပါ။


vercel.com
eimg/Sunny-Crumb: My Bakery Cafe
Sunny Crumb Bakery Café | Google AI Studio
Vercel
Find...
F
Feedback
Let's build something new
Collaborate on Pro
Enter a Git repository URL to deploy...
Continue
Import Git Repository
Select a Git provider to import an existing project from a Git Repository.
Continue with GitHub
Continue with GitLab
Continue with Bitbucket
Manage Login Connections
Clone Template
Filter
Browse All
Next.js Boilerplate
Get started with Next.js and React in seconds.
AI Chatbot
A full-featured, hackable Next.js AI chatbot built by Vercel
Express.js on Vercel
Simple Express.js + Vercel example that serves html conten...
Vite + React Starter
Vite/React site that can be deployed to Vercel
Vite + React
Home
Docs
Knowledge Base
Academy
SDKs
Help
Contact
Legal
All systems normal.

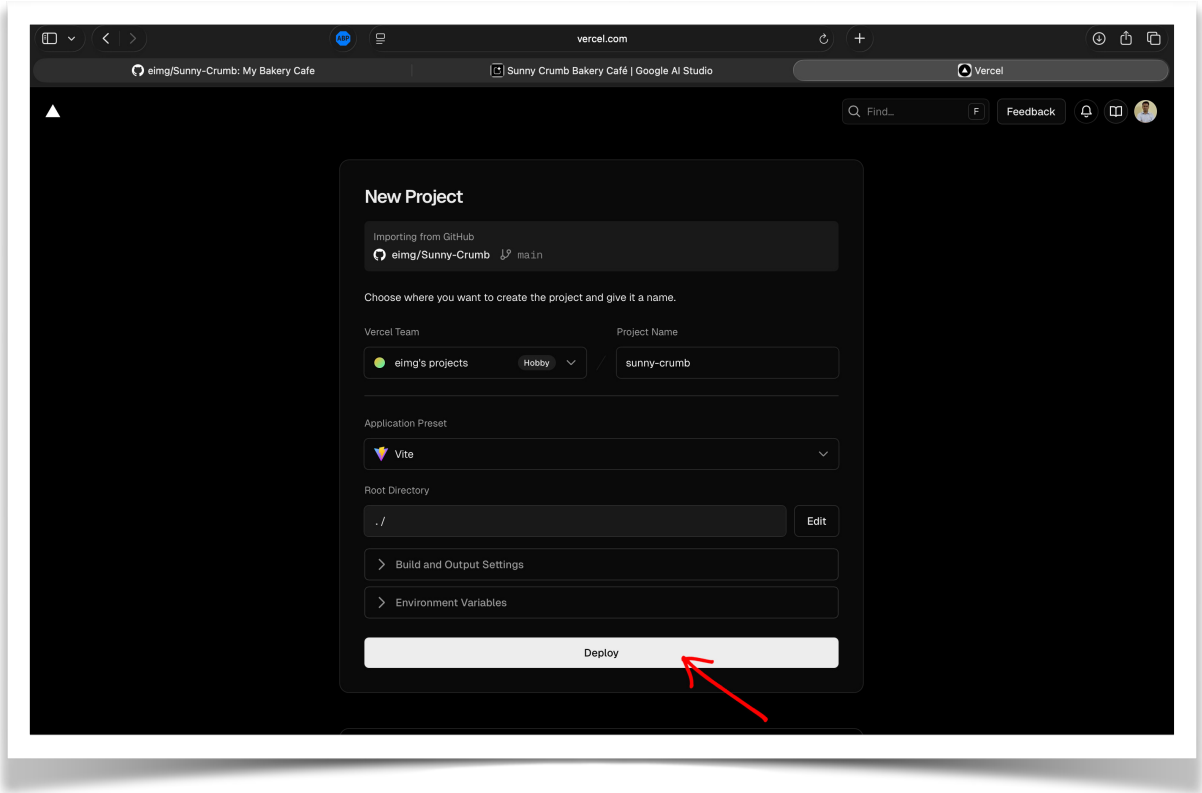

vercel.com
eimg/Sunny-Crumb: My Bakery Cafe
Sunny Crumb Bakery Café | Google AI Studio
Vercel
Find...
F
Feedback
Let's build something new
Collaborate on Pro
Enter a Git repository URL to deploy...
Continue
Import Git Repository
Select a Git Namespace
Search...
Install the GitHub application for the accounts you wish to Import from to continue
Install
Clone Template
Filter
Browse All
Next.js Boilerplate
Get started with Next.js and React in seconds.
AI Chatbot
A full-featured, hackable Next.js AI chatbot built by Vercel
Express.js on Vercel
Simple Express.js + Vercel example that serves html conten...
Vite + React Starter
Vite/React site that can be deployed to Vercel
Vite + React
Home
Docs
Knowledge Base
Academy
SDKs
Help
Contact
Legal
All systems normal.

ကိုယ့်ရဲ့ GitHub မှာတင်ထားတဲ့ ပရောဂျက်တွေပေါ်လာပါလိမ့်မယ်။ အဲ့ဒီထဲက ဖြန့်ဝေချင်တဲ့ ပရောဂျက်ကို Import လုပ်လိုက်ပါ။

အောက်က နမူနာပုံမှာကြည့်လိုက်ရင် Vercel ကအလိုအလျောက် Vite ပရောဂျက်မှန်း သိနေတာကို တွေ့ရပါလိမ့်မယ်။

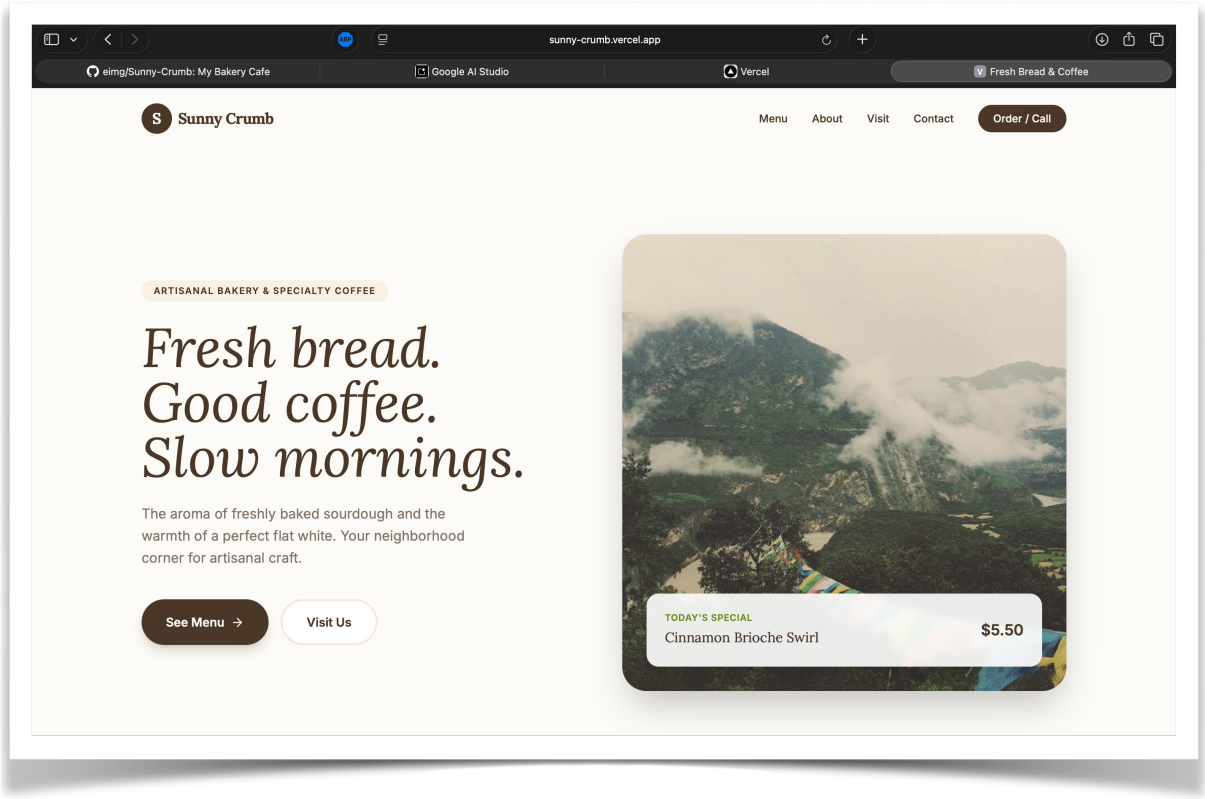


အားလုံးပြီးသွားရင် Deploy ခလုတ်ကိုနှိပ်လိုက်ရုံပါပဲ။ Vercel က ပရောဂျက်ကို **Build** လုပ်ပြီး URL တစ်ခုနဲ့ Publish လုပ်ပေးလိုက်မှာ ဖြစ်ပါတယ်။ ကိုယ်ဘာသာ လုပ်စရာမလိုပါဘူး။ Build လုပ်တာ အပါအဝင် အကုန်သူလုပ်ပေးသွားမှာပါ။



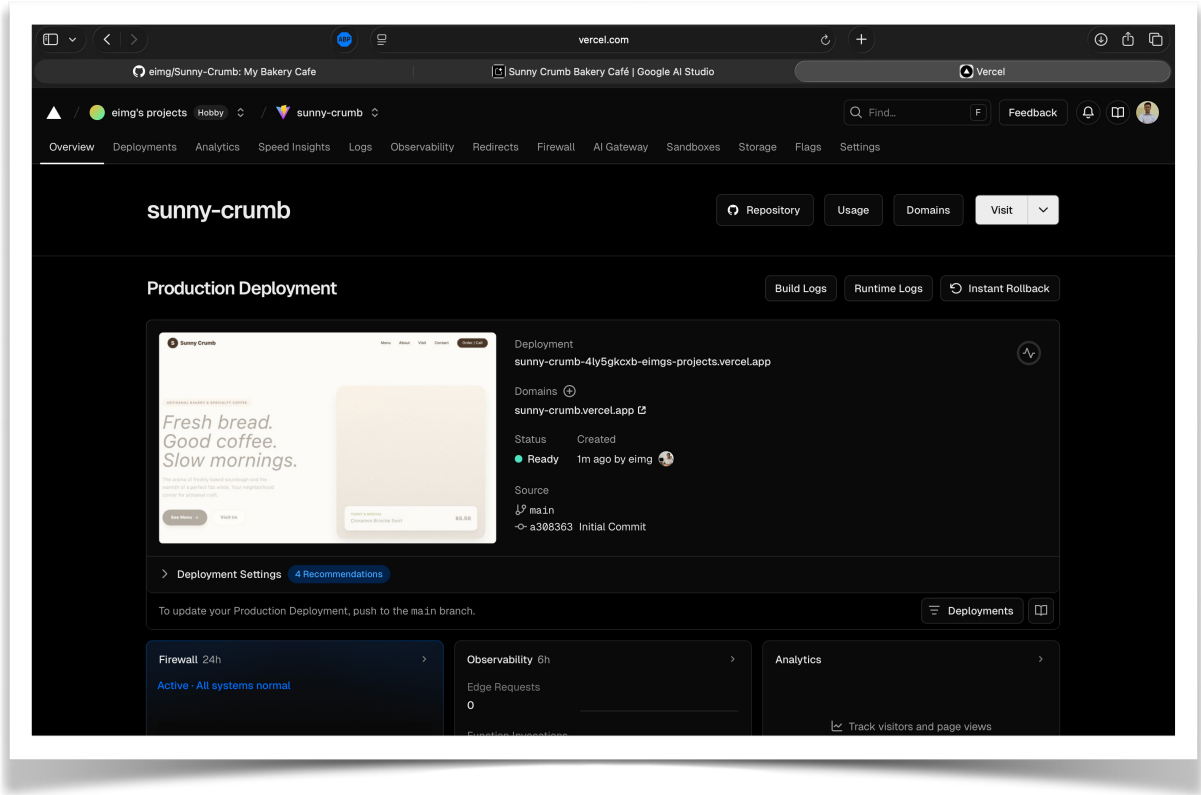
နောက်ဆုံး ရလာတဲ့လင့်နဲ့ မည်သူမဆို ဝင်ရောက်အသုံးပြုလို့ရသွားပါပြီ။ စာရေးသူ နမူနာတင်ခဲ့လိုက်တဲ့ App ကို ဒီမှာကြည့်နိုင်ပါတယ်။

https://sunny-crumb.vercel.app/



ဒါဟာ AI Studio နဲ့ Vibe Code လုပ်ထားတဲ့ ပရောဂျက်ကို Vercel နဲ့ အများအသုံးပြုလို့ ရအောင် အောင်မြင်စွာ လွှင့်တင်လိုက်နိုင်တာပါပဲ။

နောက်ပိုင်းမှာ ကိုယ့်အကောင့် Dashboard ကနေ ပရောဂျက်တစ်ခုချင်းစီအတွက် Domain Name ဝယ်ယူချိတ်တာတွေ လိုအပ်ရင် ဆက်လုပ်လို့ရပါတယ်။



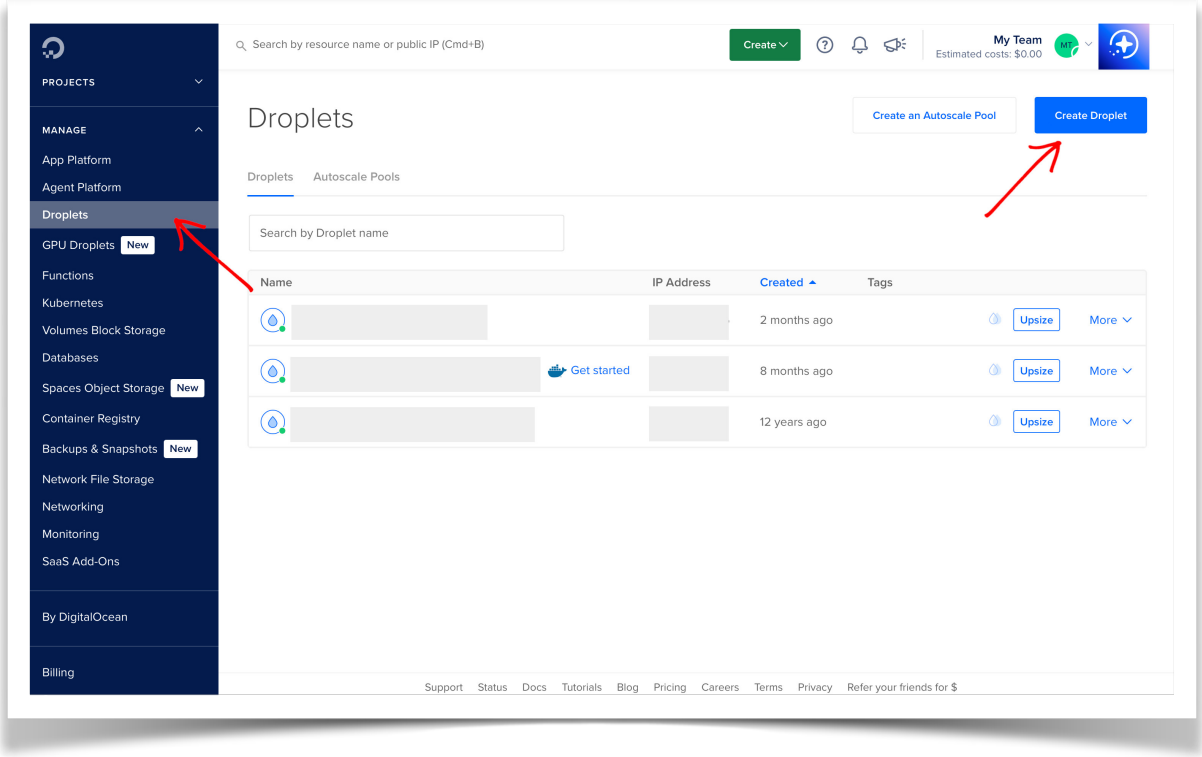
VPS

နောက်တစ်နည်းကတော့ Vercel လို ပလက်ဖောင်းမျိုးမှာ မတင်ဘဲ ကိုယ့်ဆာဗာနဲ့ကိုယ် Publish လုပ်ချင်တဲ့သူတွေအတွက် ဖြစ်ပါတယ်။ ဒီအတွက် VPS ခေါ် Virtual Private Server တွေရှိကြပါတယ်။

DigitalOcean လို့ခေါ်တဲ့ ပလက်ဖောင်းကို နမူနာ အသုံးပြုပြပါမယ်။

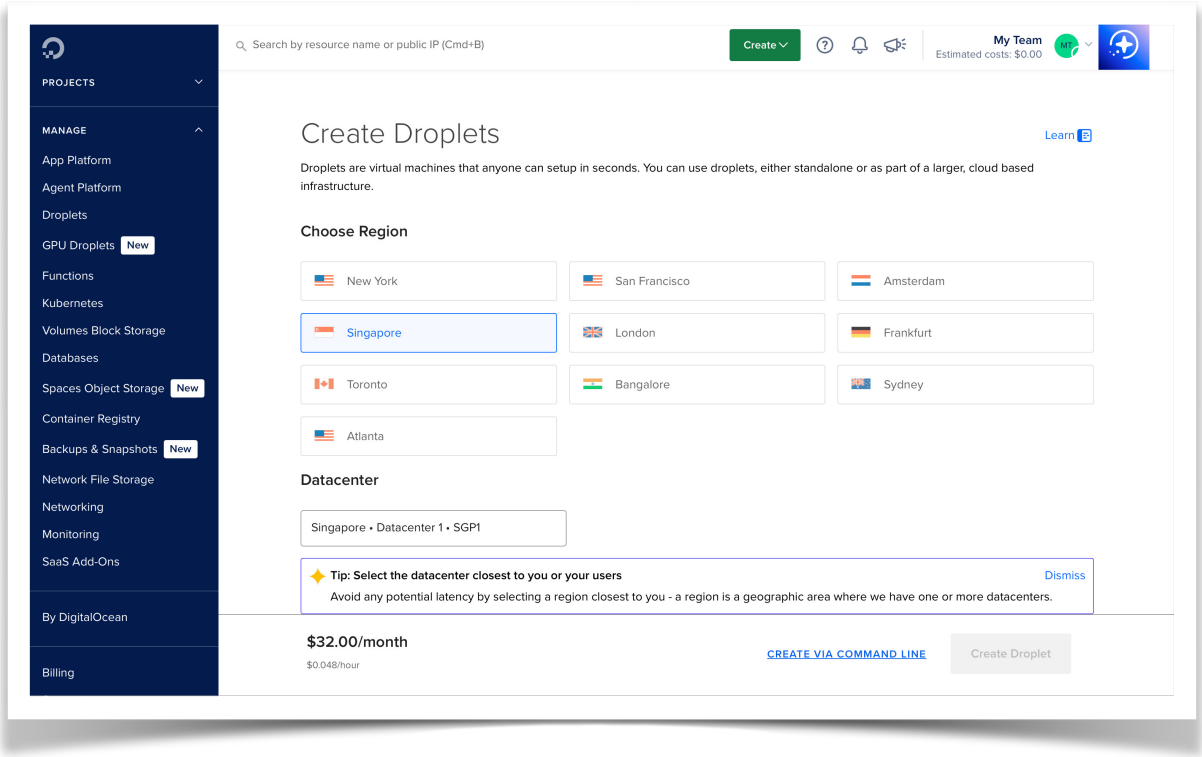
https://www.digitalocean.com/

သူကတော့ အခမဲ့မရပါဘူး။ စတင်အသုံးပြုနိုင်ဖို့အတွက် Credit Card (သို့) သူလက်ခံတဲ့ တခြားငွေပေးချေစနစ် တခုခု လိုပါတယ်။ အဲ့ဒီအပိုင်းကိုတော့ ကိုယ့်နည်းကိုယ့်ဟန်နဲ့ ဖြေရှင်းဖို့လိုပါတယ်။ အကောင့်ရပြီးနောက် ဘာတွေဆက်လုပ်ရမလဲ ပြောပြပါမယ်။

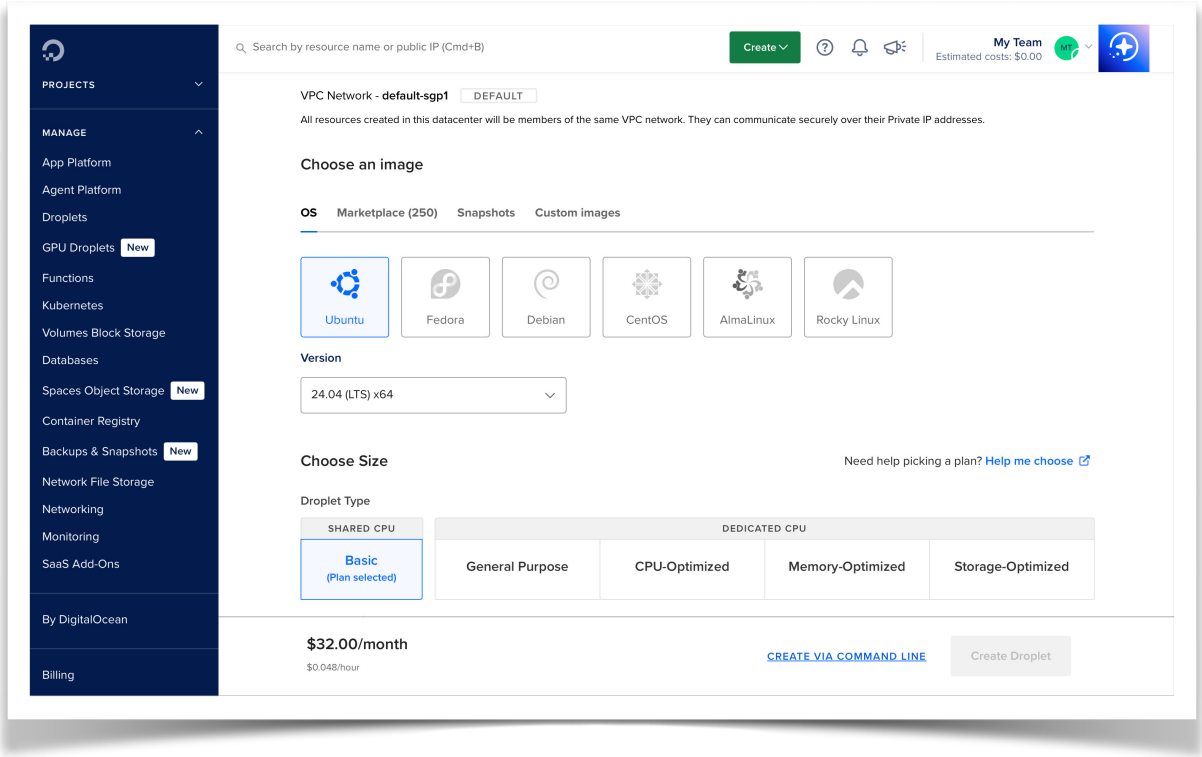


Login ဝင်လိုက်ရင် တွေ့မြင်ရမယ့် Dashboard ထဲကနေ ပုံမှာပြထားသလို **Droplets** ကို ရွေးပြီး **Create Droplet** နဲ့ VPS Server အသစ်ဖန်တီးလို့ရပါတယ်။

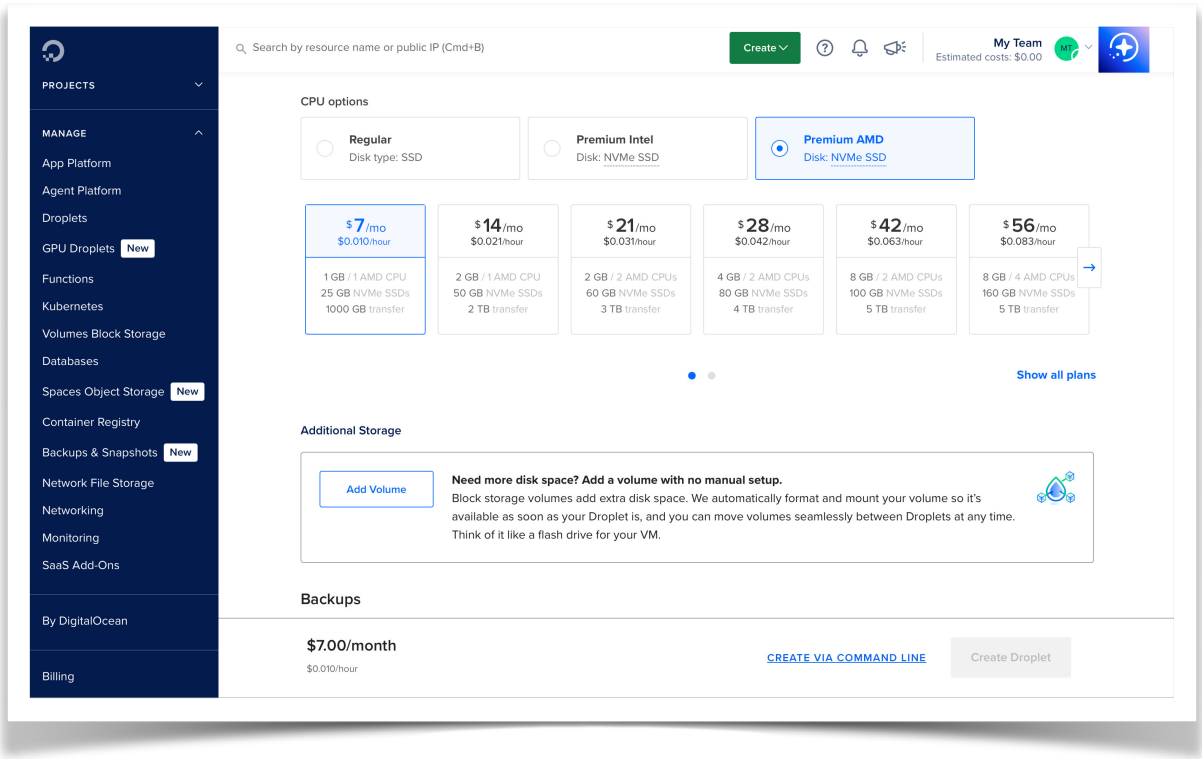
ပထမဆုံး Server တည်နေရာရွေးရပါတယ်။ ကြိုက်တာရွေးလို့ရပါတယ်။ အသုံးပြုမယ့် User တွေနဲ့ အနီးဆုံးကို ရွေးရင် သင့်တော်ပါတယ်။ နမူနာမှာ Singapore ကို ရွေးထားပါတယ်။



နောက်တစ်ဆင့်မှာ Operating System ရွေးရပါတယ်။ အားလုံးက Linux Based OS တွေချည်းပါပဲ။ အသုံးများမှာက လက်ရှိနမူနာမှာ ရွေးထားတဲ့ Ubuntu ဖြစ်ပါလိမ့်မယ်။ Ubuntu ကိုမှာ အောက်နားမှာ Version ရွေးရပါသေးတယ်။ အကြမ်းဖျဉ်းအားဖြင့် Long Term Support ခေါ် **LTS** ပါတဲ့ အမြင့်ဆုံး Version ကို ရွေးသင့်ပါတယ်။ နမူနာမှာ 24.04 LTS ကို ရွေးထားပါတယ်။



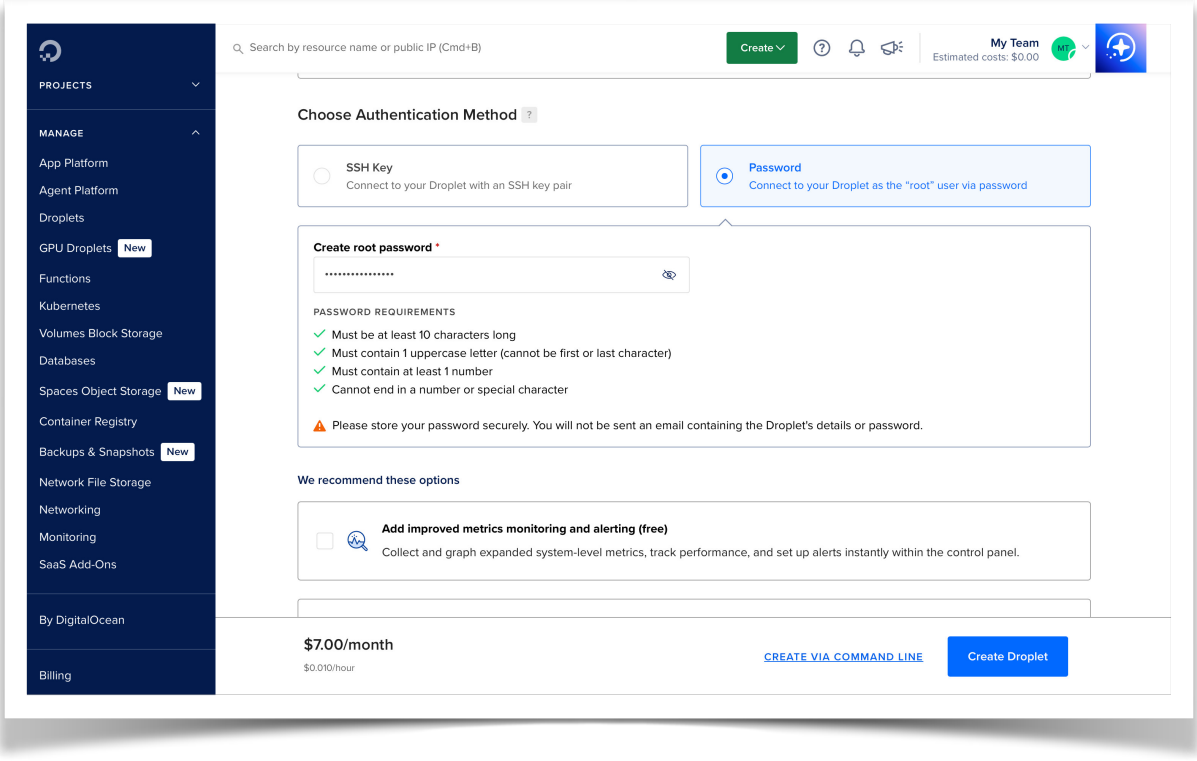
နောက်တစ်ဆင့်မှာ CPU, RAM, Storage ကိုယ်အသုံးပြုလိုတဲ့ ပမာဏကို ရွေးလို့ရပါတယ်။ နမူနာမှာ Premium AMD ထဲက ဈေးအနည်းဆုံးကို ရွေးထားပါတယ်။ ဒါကြောင့် RAM 1GB နဲ့ 25 GB SSD Storage ကိုရပါတယ်။

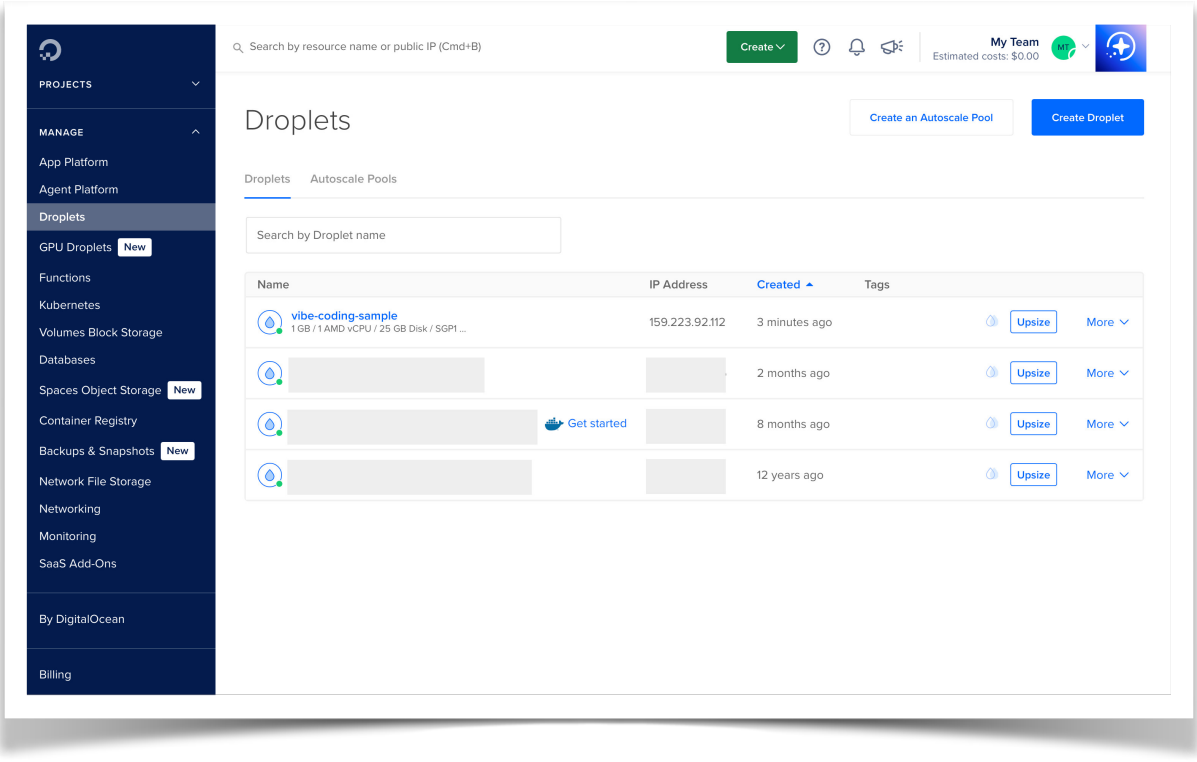


နောက်တစ်ဆင့်မှာတော့ Authentication Method ရွေးရပါတယ်။ နမူနာမှာ **Password** ကို ရွေးထားပါတယ်။ လက်တွေ့မှာ လုံခြုံရေးပိုကောင်းတာက **SSH Key** ဖြစ်ပါတယ်။ ဒီနေရာမှာတော့ ပိုလွယ်တဲ့ Password ကို အသုံးပြုပြထားပါတယ်။

စာဖတ်သူအတွက် SSH Key Gen လုပ်ရတာတွေ၊ Certificate ဖိုင်တွေ စီမံရတာတွေ အဆင်မပြေမှာစိုးလို့ပါ။ အစပိုင်း Password နဲ့ပဲစမ်းကြည့်ပြီး၊ နောက်ပိုင်းမှာ SSH Key စနစ်ကို ဘယ်လိုပြောင်းရလဲ ရှာဖွေလေ့လာပြီး ပြောင်းထားသင့်ပါတယ်။

နှစ်သက်ရာ Password တစ်ခုပေးပြီးရင် ရပါပြီ။ **Create Droplet** ကို နှိပ်လိုက်ပါ။


Search by resource name or public IP (Cmd+B)
Create
My Team
Estimated costs: $0.00
PROJECTS
MANAGE
App Platform
Agent Platform
Droplets
GPU Droplets New
Functions
Kubernetes
Volumes Block Storage
Databases
Spaces Object Storage New
Container Registry
Backups & Snapshots New
Network File Storage
Networking
Monitoring
SaaS Add-Ons
By DigitalOcean
Billing
Choose Authentication Method
SSH Key
Connect to your Droplet with an SSH key pair
Password
Connect to your Droplet as the "root" user via password
Create root password *
PASSWORD REQUIREMENTS
Must be at least 10 characters long
Must contain 1 uppercase letter (cannot be first or last character)
Must contain at least 1 number
Cannot end in a number or special character
Please store your password securely. You will not be sent an email containing the Droplet's details or password.
We recommend these options
Add improved metrics monitoring and alerting (free)
Collect and graph expanded system-level metrics, track performance, and set up alerts instantly within the control panel.
$7.00/month
$0.010/hour
CREATE VIA COMMAND LINE
Create Droplet


Search by resource name or public IP (Cmd+B)
Create
My Team
Estimated costs: $0.00
PROJECTS
MANAGE
App Platform
Agent Platform
Droplets
GPU Droplets New
Functions
Kubernetes
Volumes Block Storage
Databases
Spaces Object Storage New
Container Registry
Backups & Snapshots New
Network File Storage
Networking
Monitoring
SaaS Add-Ons
By DigitalOcean
Billing
Droplets
Create an Autoscale Pool
Create Droplet
Droplets
Autoscale Pools
Search by Droplet name
Name
IP Address
Created
Tags
vibe-coding-sample
1 GB / 1 AMD vCPU / 25 GB Disk / SGP1 ...
159.223.92.112
3 minutes ago
Upsize
More
2 months ago
Upsize
More
Get started
8 months ago
Upsize
More
12 years ago
Upsize
More

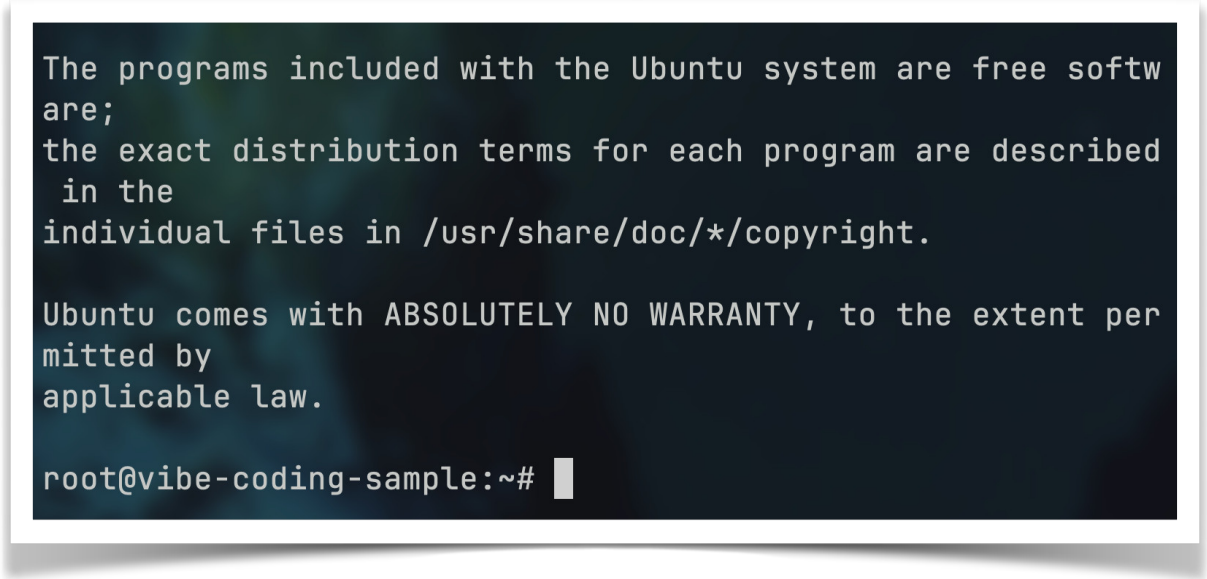
သူ Create လုပ်နေတာလေး ခဏစောင့်လိုက်ရင် နောက်ဆုံးမှာ အထက်မှာပြထားသလို ကိုယ်ပိုင် IP Address နဲ့ VPS Server တစ်ခု အသင့်ဖြစ်သွားတာကို တွေ့ရမှာ ဖြစ်ပါတယ်။

ဒီ Server ကို ကိုယ့်ကွန်ပျူတာကနေ SSH လို့ခေါ်တဲ့ နည်းပညာနဲ့ လှမ်းဝင်လို့ ရပါတယ်။ IP Address နဲ့ Password ပဲ သိဖို့လိုပါတယ်။ ဒါကြောင့် IP Address ကို နှိပ်ပြီး ကူးယူထားလိုက်ပါ။ ပြီးတဲ့အခါ Terminal ကိုဖွင့်ပြီး အခုလို ရိုက်ထည့်လိုက်ပါ။

```
ssh root@123.45.67.89
```

123.45.67.89 နေရာမှာ တကယ့် IP Address အမှန်ကို ထည့်သွင်းအသုံးပြုရမှာပါ။ ပုံမှာပြထားသလို **Are you sure you want to continue connecting** မေးခွန်း မေးလာခဲ့ရင် **yes** လို့ရိုက်ပြီး Enter နှိပ်ပေးလိုက်ပါ။

ဒါဆိုရင် Password တောင်းလာပါလိမ့်မယ်။ Server တည်ဆောက်စဉ်က ပေးခဲ့တဲ့ Password ကို ရိုက်ထည့်ပေးလိုက်ပါ။



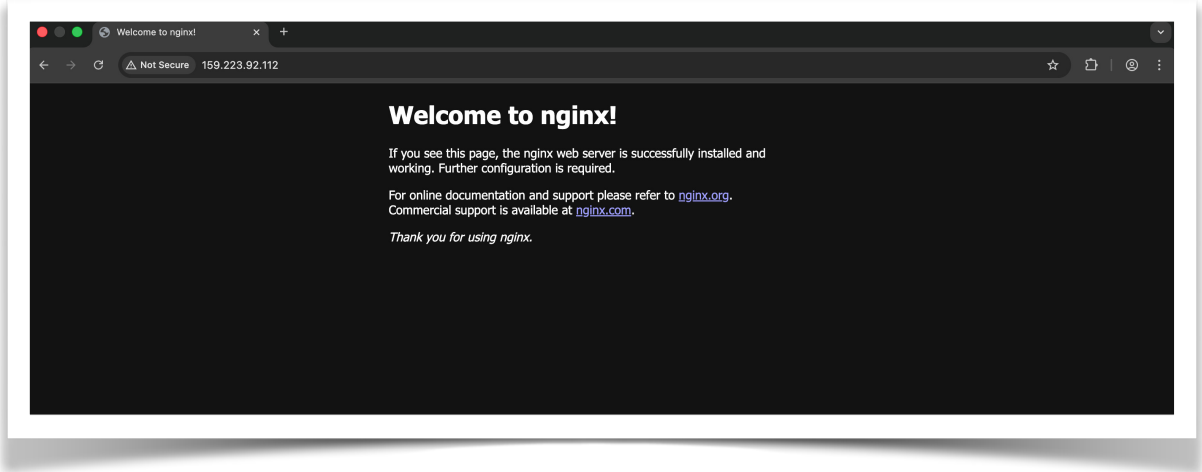
ဒါဟာ root User ခေါ်အဆင့်အမြင့်ဆုံး User အနေနဲ့ Server ထဲကို ရောက်သွားတာပါ။ တော်တော့်ကို အန္တရာယ်များပါတယ်။ မတော်တဆ ပြန်ပြင်လို့မရနိုင်တဲ့ တခုခုအမှားအယွင်းလုပ်မိတာတွေ ဖြစ်တတ်သလို၊ အကောင့်ခိုးယူခံရရင်လည်း အကုန်ပါသွားမှာပါ။

ဒါကြောင့် ခပ်ပေါ့ပေါ့သဘောမထားဖို့လိုပါတယ်။ နောက်ပိုင်းမှာ root User အကောင့်ကို ပိတ်ပြီး ရိုးရိုး User အကောင့်နဲ့ သုံးလို့ရအောင် ပြောင်းထားသင့်ပါတယ်။ ဒီနေရာမှာတော့ အဲ့ဒီ root User နဲ့ပဲ လုပ်စရာရှိတာတွေ ဆက်လုပ်လိုက်ကြပါမယ်။

ဒီ Command တွေ Run ပေးပါ။

```
apt update
apt install nginx
```

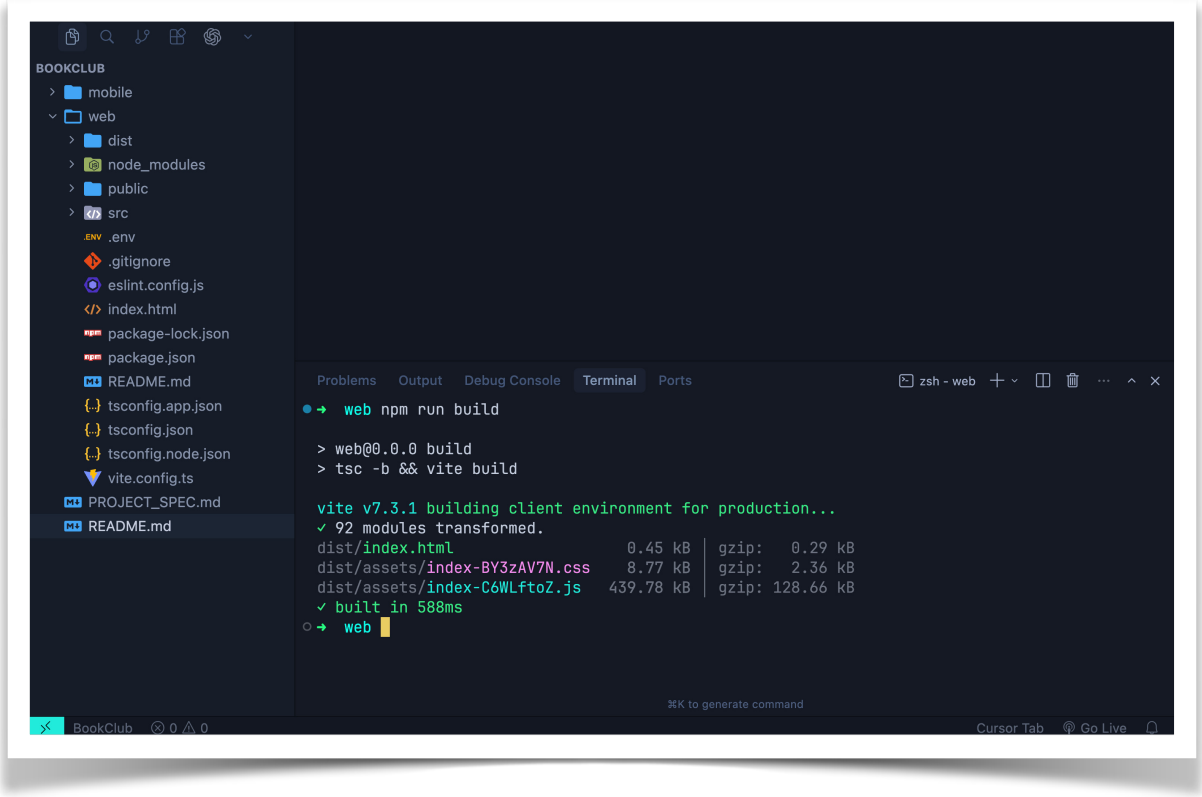
ဒါဟာ **Nginx** ခေါ် Web Server တစ်ခုကို VPS Server မှာ Install လုပ်လိုက်တာပါ။ Install လုပ်လို့ပြီးအောင် ခဏစောင့်ပြီးရင် Browser မှာ ကိုယ့် Server ရဲ့ IP Address ရိုက်ထည့်ပြီး စမ်းကြည့်လို့ရပါတယ်။ အခုလိုရလဒ်ကို ရရှိပါလိမ့်မယ်။



Nginx Web Server Run နေပြီး IP Address သိသူ မည်သူမဆိုက ကိုယ့်ဆာဗာကို ဝင်ကြည့်လို့ ရနေပါပြီ။ ဒီလောက်ဆိုရင် Ready ဖြစ်သွားပါပြီ။ ကိုယ့်ပရောဂျက်ကို တင်ကြည့်လို့ရပါပြီ။

ဒီနည်းမှာတော့ ပရောဂျက်ကို ကိုယ့်ဘာသာ **Build** လုပ်ပေးဖို့လိုပါတယ်။ Build လုပ်ပြီး ရရှိလာတဲ့ ရလဒ်ကို Server ပေါ်လှမ်းတင်ရမှာပါ။

အရင်ဆုံး Cursor Terminal မှာပဲ ဖြစ်ဖြစ် `npm run build` Command ကို Run ပြီး ပရောဂျက်ကို Build လုပ်လိုက်ပါ။ `dist` အမည်နဲ့ ရလဒ်ဖိုဒါတစ်ခုကို ရပါလိမ့်မယ်။



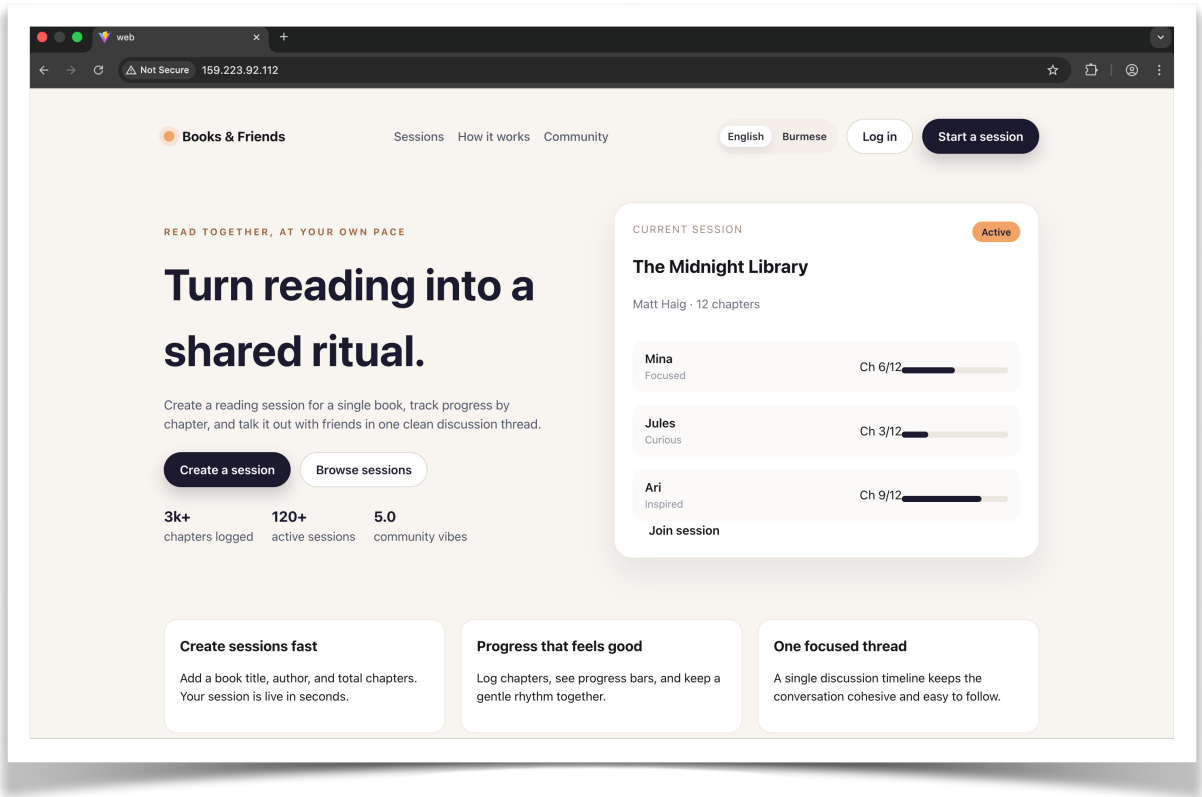
ပြီးတဲ့အခါ အဲ့ဒီ Terminal ထဲမှာပဲ အခုလို Run လိုက်ပါ။

```
scp -r ./dist/* root@159.223.92.112:/var/www/html/
```

ဒါက `scp` လို့ခေါ်တဲ့ Command ကိုသုံးပြီး `dist` ဖိုဒါထဲက ရှိသမျှဖိုင်တွေအားလုံးကို VPS Server ရဲ့ Document Root ခေါ်ပင်မ Web Server ဖိုင်ဖိုဒါထဲကို အကုန်လှမ်းကူးပို့

လိုက်တာပါ။ ဒီထက်ပိုအသေးစိတ် သိချင်ရင် ဒီ Command ရဲ့ အဓိပ္ပာယ် AI ကို မေးကြည့်လို့ ရပါတယ်။

Password တောင်းလာခဲ့ရင် ရိုက်ထည့်ပေးလိုက်ပါ။ ဒီထက်ပိုကောင်းတဲ့ တခြားနည်းတွေ ရှိသေးပေမဲ့ ဒီနည်းက အလွယ်ဆုံးပါပဲ။ ပြီးသွားပြီဆိုရင် စမ်းကြည့်လို့ရပါပြီ။



ဒါဟာ Vibe Coding လုပ်ရေးသားထားတဲ့ ပရောဂျက်ကို ကိုယ်ပိုင် VPS Server တစ်ခုနဲ့ အောင်မြင်စွာ လွှင့်တင်လိုက်ခြင်းဖြစ်ပါတယ်။

ဒီလို Frontend ပရောဂျက်တွေက လွှင့်တင်ရ လွယ်ပါတယ်။ Web Server ရှိပြီး ကိုယ့်ပရောဂျက်ဖိုင်တွေ Server ဖိုင်နေရာမှာ ထည့်ပေးလိုက်ရုံပါပဲ။ Backend ကို ကိုယ့်ဘာသာ

AI နဲ့ ရေးပြီး လွှင့်တင်ချင်ရင်တော့ ဒီထက် အများကြီး အများကြီး ပိုရှုပ်ပါလိမ့်မယ်။ Database Server တွေ Setup လုပ်ရမယ်။ Backend Server အတွက် Proxy တွေ Setup လုပ်ရပါမယ်။ အဲ့ဒါတွေတော့ ဒီနေရာမှာ ထည့်မပြောနိုင်တော့ပါဘူး။

—

အခုလိုလွှင့်တင်လိုက်တယ်ဆိုတာ စမ်းသပ်ကြည့်လိုက်တယ် လို့ပဲ သဘောထားရမှာပါ။ ဆက်လုပ်ရမှာတွေ ကျန်ပါသေးတယ်။ လက်ရှိ IP Address နဲ့ အလုပ်လုပ်တာဖြစ်လို့ သူ့အတွက် အမည် Domain Name တွေ သတ်မှတ်ပေးဖို့ လိုပါတယ်။

သိပ်မခက်ပေမဲ့ အမည်ရွေးချယ်ဝယ်ယူတာတွေထိ စာအုပ်ထဲမှာ ထည့်ပြောဖို့တော့ အခက်အခဲရှိလို့ ထည့်ပြောနိုင်မှာ မဟုတ်ပါဘူး။ Domain Name တစ်ခု နှစ်သက်ရာ နေရာကနေ ဝယ်ယူရမယ်။ ပြီးတဲ့အခါ အဲ့ဒီ Domain Name ကို ကိုယ့် VPS Server နဲ့ ချိတ်ပေးရမယ်လို့ပဲ အကြမ်းဖျဉ်းမှတ်ထားလိုက်ပါ။

Domain Name ဝယ်ယူလို့ လူကြိုက်များတာတွေထဲက တစ်ခုကတော့ Namecheap ဖြစ်ပါတယ်။ ဒီလိပ်စာမှာ လေ့လာကြည့်နိုင်ပါတယ်။

https://www.namecheap.com/

Domain Name ဝယ်ယူပြီးပြီ၊ VPS နဲ့ ချိတ်ဆက်ပြီးရင် နောက်တစ်ဆင့် ထပ်လုပ်ရမှာက **HTTPS** ကို Setup လုပ်ပေးရမှာပါ။ လုပ်ပုံလုပ်နည်းကို ဒီ Tutorial မှာ လေ့လာနိုင်ပါတယ်။

https://www.digitalocean.com/community/tutorials/how-to-secure-nginx-with-let-s-encrypt-on-ubuntu-20-04

ဒီအထိ ရသွားရင်တော့ အမှန်တကယ် အသုံးပြုလို့ရတဲ့နားကို တော်တော်နီးစပ်သွားတာပါ။ နောက်တစ်ခန်းမှာ ဆက်လက်ဖော်ပြထားတာလေးတွေ လေ့လာလိုက်ရင် ပြည့်စုံသွားပါလိမ့်မယ်။

Mobile Apps

ဆက်လက်ပြီး Mobile App တွေကို Google Play Store တို့ Apple App Store တို့မှာ ဘယ်လိုတင်ရလည်း ပြောပြပါမယ်။

ဒီအပိုင်းကတော့ ပုံသေနည်းနဲ့ အတိအကျ ပြောလို့မရနိုင်ပါဘူး။ လိုအပ်ချက်တွေ နည်းလမ်းတွေက အမြဲပြောင်းလဲ နေတတ်ပါတယ်။ ဒါကြောင့် သွားရလေ့ရှိတဲ့ အဆင့်တွေကိုပဲ ပြောပြလို့ရမှာပါ။

အဆင့် (၁)

ပထမဆုံး Google Play Store Developer Account နဲ့ Apple App Store Developer Account တွေ လိုအပ်ပါတယ်။ လက်ရှိဒီစာကို ရေးသားနေချိန်မှာ Google က (၂၅) ဒေါ်လာ ပေးရပါတယ်။ တစ်ကြိမ်ပဲပေးရပါတယ်။ Apple က တစ်နှစ်ကို ဒေါ်လာ (၁၀၀) နှစ်စဉ်ပေးရပါတယ်။ အဲ့ဒီအကောင့်တွေ ရှိမှပဲ နောက်အဆင့်တွေ ဆက်သွားလို့ ရမှာပါ။

Google Play Store - https://play.google.com/store/myplayactivity

Apple Developer Account - https://developer.apple.com/

အဆင့် (၂)

နောက်တစ်ဆင့်မှာ **Expo CLI** နဲ့ **EAS** လို့ခေါ်တဲ့ နည်းပညာတွေကို Install လုပ်ထားရမှာ ဖြစ်ပါတယ်။ သက်ဆိုင်ရာအကောင့်တွေ ရှိဖို့လိုပါတယ်။

EAS - https://docs.expo.dev/eas/

```
npm i -g eas-cli
expo login
eas login
```

EAS ဆိုတာ Expo Application Service ခေါ် React Native ပရောဂျက်တွေကို Build လုပ်ပေးနိုင်တဲ့ နည်းပညာဖြစ်ပါတယ်။ စတိုးပေါ်ထိလည်း တင်ပေးနိုင်ပါတယ်။

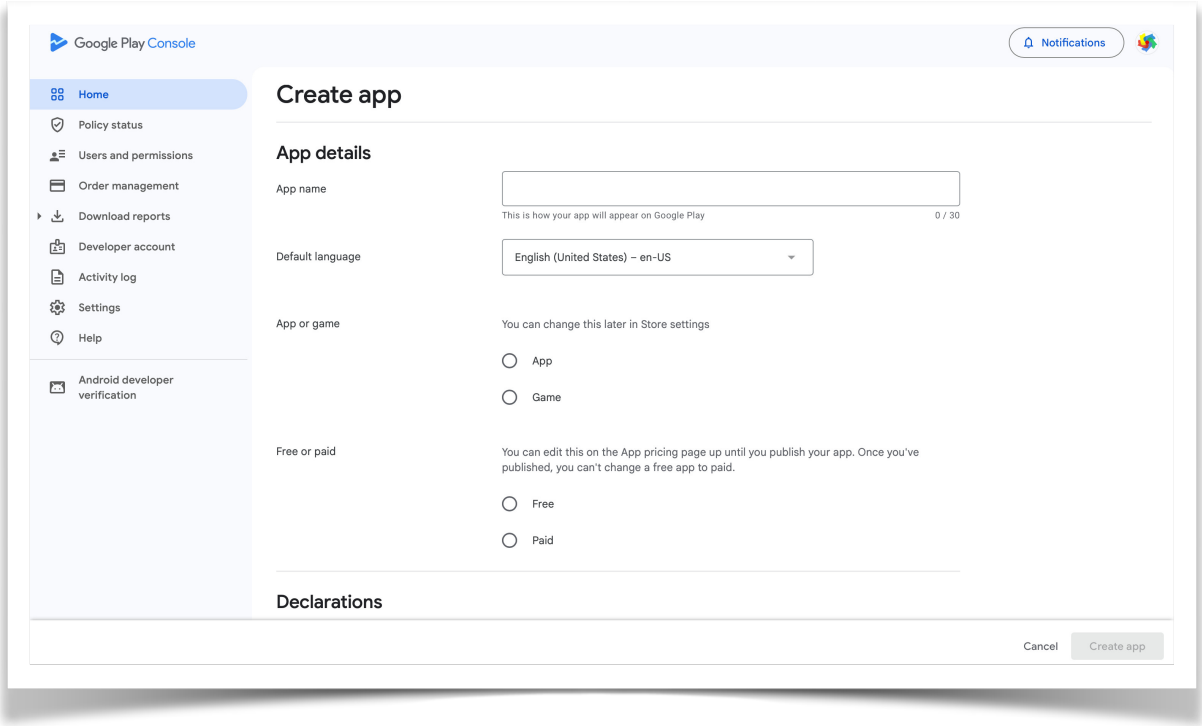
လိုအပ်တာတွေ Install လုပ်ပြီးလို့ Login တွေဝင်ပြီးရင် Configure လုပ်ပေးရပါတယ်။

```
eas build:configure
```

သူမေးတဲ့မေးခွန်းတွေကို ဖြေပေးလိုက်ပါ။ `eas.json` အမည်နဲ့ ဖိုင်တစ်ခုကို ပရောဂျက် ဖိုဒါထဲမှာ ရရှိလာပါလိမ့်မယ်။

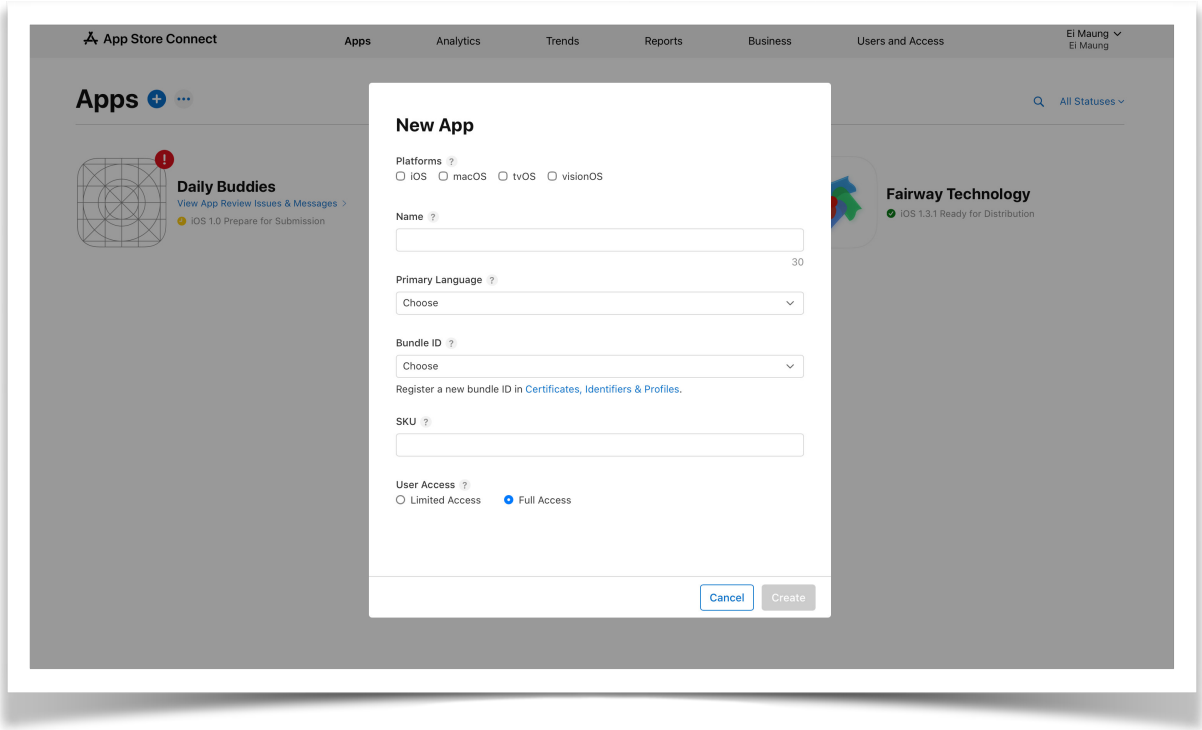
အဆင့် (၃)

Google Play Console ကိုသွားပြီး ကိုယ့် App ကို Submit လုပ်ဖို့အတွက် လိုအပ်တဲ့ ဖောင်ဖြည့်ရပါတယ်။ လင့်ထည့်ပေးမထားပါဘူး။ Google Search နဲ့ ရှာယူလိုက်ပါ။



App ဖိုင် မပါသေးဘဲနဲ့ တခြားလိုအပ်တဲ့ အချက်အလက်တွေကို ပြည့်စုံအောင် ကြိုဖြည့်ထားသင့်တာပါ။ စိတ်ရှည်ရှည်နဲ့ သူတောင်းတဲ့အချက်အလက်တွေ အကုန်စုံအောင် ဖြည့်နိုင်ဖို့လိုပါတယ်။

App Screenshot တွေ လိုလိမ့်မယ်။ Privacy Policy ဖိုင်တွေ လိုလိမ့်မယ်။ မရေးတတ်ရင် Privacy Policy ရေးပေးတဲ့ ဝက်ဘ်ဆိုက်တွေရှိပါတယ်။ ရှာပြီးရေးခိုင်းလို့ရပါတယ်။



ပြီးတဲ့အခါ Apple Store Connect ကိုလည်း သွားပြီး အလားတူ ဖောင်ဖြည့်ရပါမယ်။ ဖြည့်ပြီး **Draft** အနေနဲ့ သိမ်းထားလိုက်ပါ။ ဖြည့်ခဲ့တဲ့ **Bundle ID** ခေါ်ကိုယ့် App ရဲ့ ID ကို မှတ်ထားရပါမယ်။

အဆင့် (၄)

ပရောဂျက်ရဲ့ `app.json` ဖိုင်မှာ အမည်၊ Version စတဲ့အချက်အလက်တွေ မှန်အောင် ပြင်ပြီး အခုလို ဖုန်းမှာ Install လုပ်လို့ရတဲ့ Mobile App တွေ ဖြစ်သွားအောင် Build လုပ်ပေးလိုက်ပါ။

```
eas build --platform ios --profile production
```

```
eas build --platform android --profile production
```

သူမေးတဲ့ မေးခွန်းတွေကိုဖြေဖို့၊ သူတောင်းတဲ့ အချက်အလက်တွေကို စုံအောင် ပေးနိုင်ဖို့ လိုပါတယ်။ Build လုပ်တာ Success ဖြစ်သွားရင် App ဖိုင်တွေကို Download လုပ်လို့ရတဲ့ လင့်တွေ ပြန်ပေးပါလိမ့်မယ်။

လိုအပ်ရင် ဒေါင်းယူထားနိုင်ပါတယ်။ အတွေ့အကြုံအရ Google Play Store မှာဆိုရင် ပထမဆုံး Version ကို ကိုယ်တိုင် Manual တင်ရပါတယ်။ EAS ကို Auto တင်ခိုင်းလို့ မရပါဘူး။ နောက်ပိုင်း Version တွေ ကျတော့မှ EAS ကို Auto တင်ခိုင်းလို့ ရပါတယ်။

အဆင့် (၅)

နောက်တစ်ဆင့်မှာ EAS ကိုပဲ အခုလို Store တွေပေါ် တင်ခိုင်းလိုက်လို့ ရပါပြီ။

```
eas submit --platform ios --profile production
```

```
eas submit --platform android --profile production
```

Team ID တွေ Store အကောင့် Information တွေတောင်းလာခဲ့ရင် စုံအောင်ပေးနိုင်ဖို့လို ပါတယ်။ Google Play Store အတွက် Google Cloud Console ကနေရတဲ့ Play Console Service Account ဖိုင်လို ဖိုင်မျိုးတွေ လိုနိုင်ပါတယ်။ အဲ့ဒါတွေကိုတော့ လိုအပ်ချက်ပေါ်မူတည်ပြီး ကိုယ်တိုင် ရှာဖွေစုဆောင်းရမှာပါ။ ပုံသေနည်းနဲ့ ပြောဖို့ခက် သွားပါတယ်။

Google Play Store မှာ တင်လိုက်တဲ့ App က **Internal Testing** အနေနဲ့ အရင်ဝင်သွားပါ လိမ့်မယ်။ ကိုယ်တိုင် အရင် ကိုယ့်ဖုန်းထဲမှာ စမ်းကြည့်ရပါတယ်။ ကျေနပ်ပြီဆိုမှ Final Distribution အနေနဲ့ နောက်တစ်ဆင့် ထပ် Submit လုပ်ရပါတယ်။

Apple App Store မှာတော့ **Test Flight** လိုခေါ်တဲ့ အစမ်းတင် အစမ်းသုံးလို့ရတဲ့ စနစ်ကို ရောက်သွားပါလိမ့်မယ်။ ဖုန်းထဲမှာ Test Flight App ထည့်ထားပြီး အရင် စမ်းကြည့်ရပါ တယ်။ အဆင်ပြေပြီဆိုတော့မှ Distribution အတွက် Submit ထပ်လုပ်ရပါတယ်။

EAS က Test Version တွေထိပဲ တင်ပေးမှာပါ။ နောက်ဆုံး Final Submit တွေကိုတော့ သက်ဆိုင်ရာ ပလက်ဖောင်းပေါ်မှာ ကိုယ့်ဘာသာ လုပ်ရပါတယ်။ အဲ့ဒီအချိန်မှာ Store ဘက်က သတ်မှတ်ထားတဲ့ လိုအပ်ချက်တွေ မပြည့်စုံလို့ အကြိမ်ကြိမ် Reject ထိတာ ဖြစ် နိုင်ပါတယ်။

App ထဲမှာ အပြင် Payment ပါရင် လက်မခံတာတို့၊ Account Delete Feature မပါရင် လက်မခံတာတို့၊ Apple Login မပါရင် လက်မခံတာတို့ စသည်ဖြင့် ရစ်တာတော့ ပုံစံအမျိုးမျိုးနဲ့ ရစ်ပါတယ်။ သူတောင်းတာ အကုန်စုံအောင် ပေးနိုင်ဖို့လိုသလို၊ App မှာ လည်း သူလုပ်ခိုင်းတာတွေ စုံအောင်ထည့်ပေးဖို့လိုပါလိမ့်မယ်။

နောက်ဆုံး အားလုံးပြည့်စုံနေပြီး ဘာကြောင့်မှန်း အကြောင်းရင်းမရှင်းလင်းဘဲ Reject ထိနေတာတွေလည်း ရှိနိုင်ပါသေးတယ်။ Store ဘက်က လူတွေနဲ့ပဲ သိချင်တာတွေ မရှင်းတာတွေ ပြန်မေးပြီး ကြိုးစားကျော်ဖြတ်ရမှာပါ။ တခြားနည်းလမ်း မရှိပါဘူး။

တစ်ကြိမ် Pass ဖြစ်သွားပြီးရင်တော့ လွယ်သွားပါပြီ။ နောက်ပိုင်းတွေ သိပ်မရစ်တော့လို့ သိပ်မခက်တော့ပါဘူး။

—

Web App တွေအတွက် Deployment နဲ့ ပက်သက်ရင် အခုဒီအခန်းမှာ ဖော်ပြခဲ့တဲ့ နည်းတွေတင်ရှိတာမဟုတ်ဘဲ Vercel လိုမျိုး Cloudflare တို့ Netlify တို့လို အလားတူ နည်းပညာတွေလည်း ရှိကြပါသေးတယ်။

Cloudflare - https://www.cloudflare.com/en-gb/

Netlify - https://www.netlify.com/

DigitalOcean လိုမျိုး Linode တို့ Hetzner တို့လို နည်းပညာတွေ ရှိကြပါသေးတယ်။

Linode - https://www.linode.com/

Hetzner - https://www.hetzner.com/

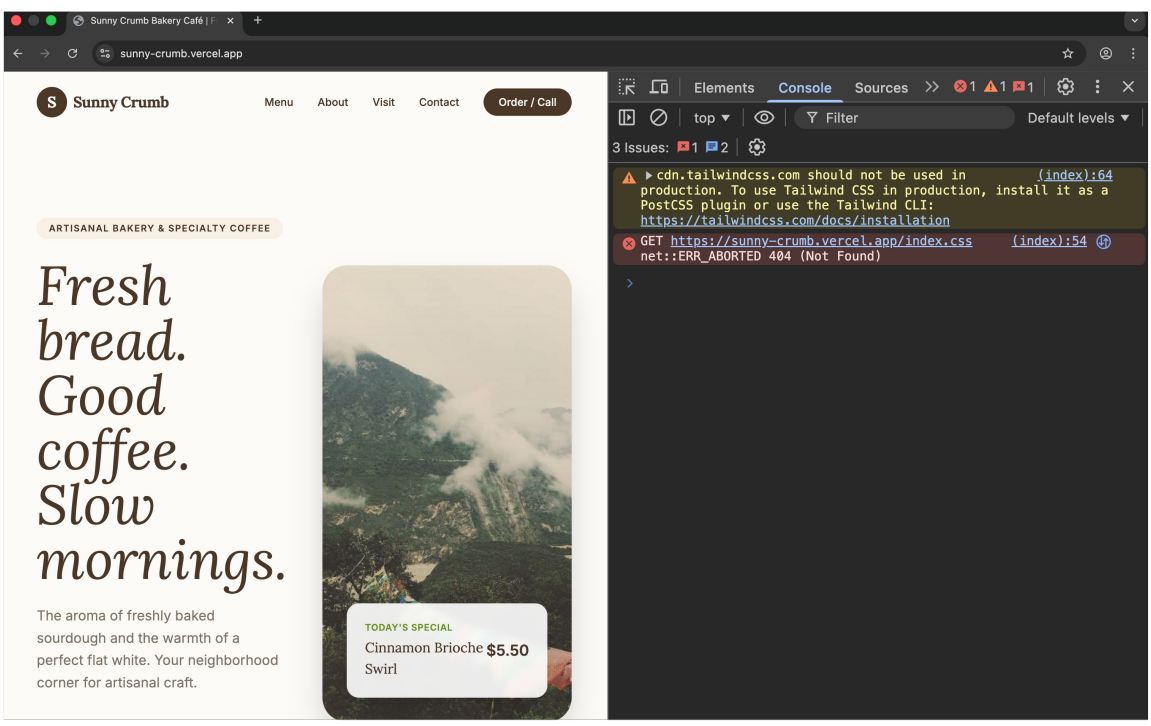
ဒါတွေတင်မက Amazon AWS တို့ Google Cloud တို့လို Cloud နည်းပညာတွေလည်း သုံးမယ်ဆိုရင် သုံးလို့ရနိုင်ပါသေးတယ်။

ဒီအခန်းမှာတော့ အတတ်နိုင်ဆုံး အလွယ်ဆုံးနဲ့ အမြန်ဆုံး နည်းတွေကို ရွေးချယ်ဖော်ပြခဲ့တာပါ။ လက်တွေ့မှာ ရေးပြီးရင် လွှင့်တင်လိုက်တယ်ဆိုရုံနဲ့ မလုံလောက်ပါဘူး။ နောက်ဆက်တွဲ သတိပြုစရာတွေ အများကြီး ကျန်ပါသေးတယ်။

အဲ့ဒီအကြောင်းတွေကို နောက်တစ်ခန်းမှာ ဆက်လက်ဖော်ပြပါမယ်။

အခန်း (၉) - သတိပြုစရာများ

ပြီးခဲ့တဲ့အခန်းမှာ Vercel နဲ့ လွှင့်တင်လိုက်တဲ့ App ကို Browser နဲ့ဖွင့်ထားစဉ် Console ဖွင့်ကြည့်လိုက်ရင် အခုလို Error တချို့တွေ့ရနိုင်ပါတယ်။



အနီရောင်နဲ့ ပြနေတဲ့ 404 Error က ပိုလန့်ချင်စရာဖြစ်နေပေမဲ့ တကယ်တော့ အဝါရောင်နဲ့ ပြနေတဲ့ Warning ကတောင် ပြဿနာပိုကြီးပါသေးတယ်။

“**cdn.tailwindcss.com should not be used in production**” လို့ ပြောထားပါ တယ်။ ဒါဘာကိုပြောတာလဲ။ ဘယ်လိုဖြေရှင်းရမှာလဲဆိုတာ သက်ဆိုင်ရာ Coding နည်း ပညာ လေ့လာထားသူတွေက တန်းသိနိုင်ပါတယ်။

Coding နည်းပညာတွေ မသိရင်တော့ ဒါကိုပြဿနာလို့တောင် ထင်ချင်မှ ထင်ပါလိမ့် မယ်။ ကိုယ့် App ကို ဖြစ်သင့်တာထက် အများကြီးပိုနှေး နေစေတဲ့ ပြဿနာပါ။ ထုံးစံ အတိုင်း အဲ့ဒီ Message တွေကို ကူးယူပြီး AI ကို ပေးလိုက်လို့ရပါတယ်။ ဒါဆိုရင် ဖြေ ရှင်းပေးပါလိမ့်မယ်။

AI က ခိုင်းသမျှလုပ်နိုင်ပါတယ်။ ဒါပေမဲ့ကိုယ်က ဘာခိုင်းရမလဲဆိုတာ သိဖို့တော့လိုပါ တယ်။ အဲ့ဒီမှာ Coding ပိုင်းကျွမ်းကျင်မှုတွေက စကားပြောလာတာပါ။ ကျွမ်းကျင်သူ တွေက ဘာကိုစစ်ရမလဲ၊ ဘာလုပ်ခိုင်းရမလဲ၊ ဘယ်လိုခိုင်းရမလဲဆိုတာ ပိုသိကြပါတယ်။

အခုဒီစာအုပ်မှာ တွေ့ခဲ့ကြပါပြီ။ Coding မသိလည်း AI နဲ့ App တွေ ဖန်တီးလို့ရပါတယ်။ Publish လုပ်လို့ရပါတယ်။ ဒါပေမဲ့ အကန့်အသတ်တစ်ခု ရှိပါလိမ့်မယ်။ Coding သိမှသာ ဒီအကန့်အသတ်တွေကို ကျော်ပြီး အဆင့်အတန်းမြင့်တဲ့ App တွေထိ လုပ်နိုင်မှာ ဖြစ်ပါ တယ်။

ဒီနေရာမှာ တစ်ထိုင်တည်းနဲ့ အကုန်သိသွားအောင် ပြောပြဖို့မဖြစ်နိုင်ပေမဲ့၊ ပရောဂျက်တစ်ခုကို လွှင့်တင်ပြီးနောက် စစ်ဆေးသင့်တာတွေကို စာရင်းပြု ထည့်သွင်းဖော်ပြပေးချင်ပါတယ်။

၁။ လျှို့ဝှက်အချက်အလက်များ

API Key နဲ့ Password အပါအဝင် အသုံးပြုသူ မသိသင့်တဲ့ အချက်အလက်တွေဟာ Frontend ကုဒ်တွေထဲမှာ လုံးဝ ထည့်မထားရပါဘူး။ Vibe Code လုပ်သူတွေဟာ ဒါကို မသိဘဲ ထည့်ထားမိတတ်ကြပါတယ်။

Frontend ကုဒ်တွေရဲ့ အလုပ်လုပ်ပုံက ဒီလိုပါ။ အသုံးပြုသူ User ရဲ့ Device ဆီကို ကုဒ်တွေ ပေးပို့ပါတယ်။ အသုံးပြုသူဆီကို ရောက်တော့မှ ကုဒ်တွေက အလုပ်လုပ်ပါတယ်။ ဒီတော့ ဆိုကြပါစို့ ကုဒ်ထဲမှာသာ Admin Password ကို ထည့်ထားတယ်ဆိုရင် ကုဒ်ကိုဖွင့်ကြည့်တတ်တဲ့ User က Admin Password ကို သိသွားမှာပါ။

တခြားကိုယ်အခပေးဝယ်ယူထားတဲ့ API Key လိုဟာတွေ ထည့်ထားမိရင်လည်း အတူတူပါပဲ။ User က ကုဒ်ကိုဖွင့်ကြည့်လို့ တွေ့သွားရင် အဲ့ဒီ ကိုယ့်ကိုယ်ပိုင် API Key ကို User က ယူသုံးလို့ရသွားမှာပါ။

တချို့လည်း .env လို့ခေါ်တဲ့ ဖိုင်တစ်ခုနဲ့ အဲ့ဒီလို လျှို့ဝှက်အချက်အလက်တွေကို ထားရတယ်။ ကုဒ်ထဲမှာ မထားရဘူးလို့ ကြားဖူးနိုင်ပါတယ်။ အဲ့ဒါက Backend အတွက်ပါ။ Frontend ကတော့ .env ထဲမှာထားလည်း အချက်အလက်တွေက မလုံခြုံပါဘူး။

မဖြစ်မနေကုဒ်ထဲမှာ အဲ့ဒီလို လျှို့ဝှက်အချက်အလက်တွေ အသုံးပြုဖို့လိုရင် Backend ဘက်မှာမှာပဲထားပြီး Frontend က အဲ့ဒီ Backend ကို ခေါ်ယူအသုံးပြုတဲ့နည်းကို အသုံးပြုနိုင်ပါတယ်။

ရှေ့ပိုင်းက နမူနာတစ်ခုမှာ Frontend ကုဒ်ကို .env နဲ့ AI က ကုဒ်တချို့ရေးပေးသွားတာ မှတ်မိကြပါလိမ့်မယ်။ အဲ့ဒီကုဒ်မှာတုံးက ပြဿနာမရှိပါဘူး။ ဘာဖြစ်လို့လဲဆိုတော့ .env ထဲမှာ သိမ်းသွားတာက အများကို ဖြန့်ဝေလို့ရတဲ့ Supabase URL နဲ့ Publishable Key တွေကို ထည့်ခဲ့တာ မို့လို့ပါ။ အကယ်၍သာ Secret Key တွေဆိုရင် ထည့်လို့မရပါဘူး။

ဒီလိုပြဿနာမျိုးကို ရှောင်ရှားဖို့အတွက် ထုံးစံအတိုင်း AI ကို ရေးထားတဲ့ကုဒ်အတွက် Security Review လုပ်ခိုင်းလို့ရပါတယ်။ ဒါဆိုရင်သူက ကုဒ်တွေကို လိုက်ကြည့်ပြီး အဲ့ဒီလိုပြဿနာမျိုး ရှိနေရင် ပြန်ပြောပါလိမ့်မယ်။ ဖြေရှင်းနည်းတွေ အကြံပေးပါလိမ့်မယ်။ အထောက်အကူဖြစ်ပါတယ်။

ဒါပေမဲ့ ထပ်ဆင့်သတိပြုရမှာက AI က မတွေ့လိုက်ဘဲ ကျန်သွားတာမျိုးတွေလည်း ရှိတတ်ပါတယ်။ ဒါကြောင့် ကိုယ့်ဘက်ကလည်း ဒီပြဿနာမျိုး မရှိအောင် သတိထားပေးဖို့လိုအပ်ပါလိမ့်မယ်။

၂။ Database Indexes

Database ထဲက ဒေတာတွေကို ရယူတဲ့အခါ လျှင်မြန်စေဖို့အတွက် သင့်တော်တဲ့ Index တွေ သတ်မှတ်ထားရပါတယ်။ Index ဆိုတာ စာအုပ်တစ်အုပ်ရဲ့ မာတိကာလိုပါပဲ။ လိုချင်တာ သိချင်တာကို Index မှာ အရင်ကြည့်ပြီး ချက်ချင်းဆွဲထုတ်ယူလို့ရပါတယ်။

AI ပဲ ဖြစ်ဖြစ်၊ လူပဲဖြစ်ဖြစ်၊ အစပိုင်းမှာ လိုချင်တဲ့ ရလဒ်ရဖို့ကိုပဲ ဦးစားပေးကြပါတယ်။ မြန်ဖို့တွေဘာတွေ ထည့်မစဉ်းစားကြသေးပါဘူး။ ပရောဂျက်ပြီးသွားတဲ့အခါ အဲ့ဒါကို မေ့သွားပြီး Publish လုပ်လိုက်တာ အလုပ်တော့ လုပ်နေတယ်။ မြန်သင့်သလောက် မမြန်ဘူးဖြစ်နေတယ်ဆိုတာ ဖြစ်နိုင်ပါတယ်။ အထူးသဖြင့် အချက်အလက်တွေ များလာတဲ့အခါ ပိုပြီးတော့ သိသာလာမှာပါ။

ထုံးစံအတိုင်း ကိုယ်တိုင်လုပ်စရာမလိုပါဘူး။ Database Index တွေ ကြည့်ပြီး သတ်မှတ်ပေးပါ လို့ AI ကို ခိုင်းလိုက်ဖို့ပဲ လိုပါတယ်။ ဒါဆိုရင် AI က သင့်တော်တဲ့ Index တွေ သူကိုယ်တိုင်ထည့်လို့ရရင် ထည့်ပေးပြီး မရရင် ဘယ်လိုထည့်ရမလဲ ပြောပြပါလိမ့်မယ်။

၃။ Query Performance

ဒါက အပေါ်ကအချက်နဲ့ အတူတူပါပဲ။ အများအားဖြင့် အစပိုင်းလိုချင်တဲ့ရလဒ် ရဖို့ကိုပဲ အဓိကထားပြီး လုပ်ကြလို့ Loop ထဲမှာ Query တွေရေးထားမိပြီး ထပ်ခါထပ်ခါ မ Run သင့်တာတွေကို Run မိနေတာမျိုး ဖြစ်တတ်ပါတယ်။ AI လည်း ဒါမျိုးတွေ လုပ်ပါတယ်။

ဒါကြောင့် ပရောဂျက် Feature တွေတော့ လိုချင်တာရသွားပြီ၊ Query Performance ကောင်းအောင် ဆက်လုပ်ပါ ဆိုတာမျိုးကို ကိုယ့်ဘက်က သီးခြားခိုင်းပေးဖို့ လိုနိုင်ပါတယ်။ N+1 Query ခေါ် ထပ်ခါထပ်ခါ မခေါ်သင့်ဘဲ ခေါ်ထားမိတဲ့ Query အပါအဝင် တခြား ဒေတာတွေ မလိုဘဲယူထားမိတဲ့ Query တွေကို မြန်သွားအောင် Optimize လုပ်ပေးပါလိမ့်မယ်။

၄။ Database Security

တချို့ Database တွေဟာ ရေးသားစဉ်မှာ Password တွေဘာတွေ မပါဘဲ အလွယ်တကူ အသုံးပြုအောင် ဒီတိုင်းအလွတ် ထားကြလေ့ရှိပါတယ်။ ပြီးသွားလို့ လွှင့်တင်တဲ့အခါ လုံခြုံတဲ့ Password မပေးမိလို့ ရှေ့က App က အလုပ်လုပ်နေပေမဲ့ နောက်က Database က ဟောင်းလောင်း ဖြစ်နေတာတွေလည်း ရှိကြပါတယ်။

ဒီစာအုပ်ထဲက နမူနာတွေမှာတော့ ကိုယ်ပိုင် Database နဲ့ မသွားဘဲ Supabase ကို သုံးခဲ့ကြပါတယ်။ Supabase မှာ RLS ခေါ် Row Level Security ဆိုတဲ့ သဘောတရား ရှိပါတယ်။ Default က အဲ့ဒီလုံခြုံရေးကို အသုံးမပြုဘဲ ပိတ်ထားပါတယ်။ ကိုယ်က အသုံးပြုပေးဖို့ မေ့နေရင် Database ကို မည်သူမဆို ဝင်လို့ရနေတဲ့သဘောပါ။

ဒါကြောင့် AI ကိုပဲ RLS Security တွေ အကုန်သုံးဖို့၊သင့်တော်တဲ့ လုံခြုံရေးအစီအမံနဲ့ အလုပ်လုပ်အောင် ကုဒ်တွေကို ရေးပေးဖို့ ကိုယ့်ဘက်က ပြောပေးဖို့လိုအပ်ပါလိမ့်မယ်။

၅။ File Security

အလားတူပါပဲ။ ဖိုင်တွေသိမ်းတဲ့ ဖိုဒါတွေကို အပြင်လူဝင်လို့မရအောင် ပိတ်လိုက်ဖို့ မေ့နေတာတွေ ရှိတတ်ပါတယ်။ တချို့ဖိုင်တွေကတော့ ပြဿနာမရှိပါဘူး။ သူ့သဘာဝအရကို က Public ဖြစ်ဖို့ လိုတာတွေ ရှိနိုင်ပါတယ်။ ဥပမာ - Profile Photo လို ဟာမျိုးတွေပါ။ အဲ့ဒီလို Public ဖြစ်ဖို့လိုတဲ့ ဖိုင်တွေရှိသလို၊ သက်ဆိုင်သူမှသာ ကြည့်ခွင့်ရှိသင့်တဲ့ ဖိုင်တွေလည်း ရှိပါတယ်။ ဥပမာ - Payment Record လိုဟာမျိုးတွေပါ။ အဲ့ဒီလို အရေးကြီးဖိုင်တွေရှိနေတဲ့ ဖိုဒါတွေကို အပြင်လူဝင်လို့မရအောင် ထည့်ပိတ်ဖို့လည်းလိုပါတယ်။

Supabase ရဲ့ Bucket တွေကို နမူနာမှာ သုံးခဲ့ပါတယ်။ ပေးခဲ့တဲ့ နမူနာကတော့ Public ပဲ ပေးခဲ့ပေမဲ့ ဖြစ်နိုင်ရင် Private Bucket တွေပဲ သုံးသင့်တယ် ဆိုတာလေး သတိပြုပါ။

၆။ UI Performance

Vibe Code လုပ်တဲ့ပရောဂျက်တွေရဲ့ ၉၀% လောက်က ဒီစာအုပ်ထဲက နမူနာတွေမှာ အသုံးပြုခဲ့တဲ့ React ကိုပဲ အသုံးပြုကြတာ ဖြစ်လိမ့်မယ်လို့ ယူဆပါတယ်။ React မှာ အဖြစ်များတဲ့ ပြဿနာတချို့ ရှိပါတယ်။ အဲ့ဒါတွေကတော့ Re-Render နဲ့ Fetch Waterfall တို့ပါ။

Re-Render ဆိုတာ Data မှာ အပြောင်းအလဲတခုခု ရှိတဲ့အခါ UI ကို အလိုအလျောက် ပြောင်းလဲ ဖော်ပြပေးတဲ့လုပ်ဆောင်ချက် ဖြစ်ပါတယ်။ ပြဿနာက တစ်ခုပြောင်းတာကို

လိုလည်းမလိုဘဲ အများကြီး Re-Render ပြန်လုပ်နေတာတွေ၊ ရေးသားပုံစနစ်မကျလို့ မလုပ်သင့်ဘဲ Re-Render ပြန်လုပ်နေတာတွေ ရှိတတ်တာပါ။

Fetch Waterfall ဆိုတာကတော့ Frontend က Backend တွေ Service တွေနဲ့ ဆက်သွယ်တဲ့အခါ ပြိုင်တူဆက်သွယ် အလုပ်လုပ်ရင် ရနိုင်လျက်နဲ့ တစ်ခုပြီးမှတစ်ခု အစီအစဉ်အတိုင်း ဆက်သွယ်အလုပ်လုပ်နေတဲ့ အခြေအနေမျိုးပါ။ သူလည်း အများအားဖြင့် ရေးသားပုံစနစ်မကျရင် ဖြစ်တတ်တာပါပဲ။

ဒါမျိုးတွေကြောင့် UI ရဲ့ စွမ်းဆောင်ရည်ဟာ ရသင့်တာထက် အများကြီး ပိုနှေးနေတာတွေ ဖြစ်နေနိုင်ပါတယ်။

AI ရဲ့ ပြဿနာက သူ့ကို Train ထားတဲ့ နမူနာကုဒ်တွေထဲမှာ အဆင့်မြင့်စနစ်ကျတာတွေပါသလို၊ စနစ်မကျတဲ့ ရေးနည်းတွေလည်း ပါနေနိုင်ခြင်းဖြစ်ပါတယ်။ ဒီတော့ သူက Smart ဖြစ်တယ် ရေးပေးနိုင်တယ်ဆိုတိုင်း အမြဲတမ်း အကောင်းဆုံးရေးနည်းကို ရတာ မဟုတ်နိုင်ပါဘူး။ ပိုကောင်းတဲ့ ရေးနည်းတွေ သုံးပေးဖို့ နောက်ကနေ ပြန်လိုက်ပြင်ခိုင်းရတာတွေ ရှိတာ ထုံးစံပါပဲ။

UI ပိုင်းနဲ့ ပတ်သက်ရင် မလိုအပ်တဲ့ Re-Render နဲ့ Fetch Waterfall တို့ဟာ AI ကို မဖြစ်မနေ ပြန်စစ်ပြီး ပြန်ပြင်ခိုင်းသင့်တဲ့ကိစ္စတွေပါ။

၇။ Abuse

ကိုယ့်ပရောဂျက်မှာ Register ဖောင်တစ်ခုရှိတယ်ဆိုကြပါစို့။ အနှောက်အယှက်ပေးလိုသူက တကယ် Register လုပ်လိုတာမဟုတ်ဘဲ Bot တွေ Run ပြီးတော့ Spam အကောင့်ပေါင်းများစွာ လာထည့်ရင် ဘယ်လိုလုပ်မလဲ။

Public အင်တာနက်မှာ လင့်တစ်ခုနဲ့ ဝင်လို့ရပြီဆိုတာနဲ့ ဒီလို Abuse တွေက လာမှာပါ။ Vercel တို့ Supabase တို့ကို အသုံးပြုထားရင် သူတို့ဘက်ကလည်း သင့်တင့်တဲ့ အကာအကွယ်တွေတော့ တစ်ခါတည်း လုပ်ပေးထားကြပါတယ်။

ကိုယ့်ဘက်ကလည်း **reCAPTCHA** တို့ **API Rate Limit** တို့လို နည်းစနစ်တွေနဲ့ သင့်တင့်တဲ့ အကာအကွယ်ထည့်ယူဖို့လိုပါတယ်။ ဒီနေရာမှာ ဒီနည်းပညာတွေကို အကျယ်မချဲ့နိုင်ပါဘူး။ စာဖတ်သူက ဒီလိုသဘောတရားမျိုးတွေ ရှိတာကို သိထားပြီး AI နဲ့ တိုင်ပင်ဆောင်ရွက်နိုင်အောင်ပဲ ရည်ရွယ်ပြီး ထည့်ပြောနိုင်ပါတယ်။

၈။ Backup

ကိုယ်ကိုယ်တိုင်ရဲ့ အမှားကြောင့်ပဲဖြစ်ဖြစ်၊ AI ရဲ့ အမှားကြောင့်ပဲဖြစ်ဖြစ်၊ User ရဲ့ အမှားကြောင့်ပဲဖြစ်ဖြစ်၊ အချက်အလက်တွေ ပျောက်သွားတယ်၊ ပျက်သွားတယ်၊ ဆုံးရှုံးသွားတယ်ဆိုတာ ရှိတတ်ပါတယ်။ တစ်နည်းမဟုတ်တစ်နည်းနဲ့ ရှိလာမှာပါပဲ။

ဒါကြောင့် ကုဒ်တွေ၊ ဖိုင်တွေ၊ ဒေတာတွေကို ဘယ်လို Backup လုပ်မှာလည်းဆိုတာ ထည့်စဉ်းစားထားဖို့ လိုပါတယ်။ AI ကို Backup Script တွေ ရေးခိုင်းလို့ ရပါတယ်။ သူရေးပေးတဲ့ Script ကို Run လိုက်ရုံနဲ့ လိုအပ်တဲ့ Backup တွေကို တခြားစိတ်ချရတဲ့ တစ်နေရာမှာ ပို့သိမ်းသွားတယ်ဆိုတာမျိုး ဖြစ်အောင် လုပ်ထားရပါတယ်။

၉။ Monitoring

ဘယ် Page တွေမှာ Error အဖြစ်များလဲ။ ဘယ်အချိန်ဆိုရင် အသုံးပြုသူများလဲ။ ပြဿနာတခုခု တက်သွားတယ်၊ ဘာကြောင့်တက်သွားလဲ မသိဘူး၊ အဖြေရှာဖို့အတွက် Error မှတ်တမ်းတွေ သိချင်တယ်။ စသည်ဖြင့် လိုအပ်တတ်ပါတယ်။

Vercel တို့ Supabase တို့မှာ သင့်တင့်တဲ့ Activities Logs တွေ Usage Logs တွေ ပါဝင်ကြပါတယ်။ အဲ့ဒါတွေကို အသုံးချဖို့လိုပါတယ်။ AI ကိုလည်း Logging System ထည့်လုပ်ခိုင်းလို့ရပါတယ်။ ကိုယ့် App ထဲမှာ ကိုယ်တိုင် လိုအပ်ရင် ဝင်ကြည့်လို့ရအောင်ပါ။

အဲ့ဒါတွေမရှိရင် တခုခုအဆင်မပြေတဲ့အခါ ဘာကြောင့်အဆင်မပြေတာလဲ ဘယ်မှာသွားကြည့်ရမှန်းမသိဘူး ဖြစ်နေတတ်ပါတယ်။

—

ဒါတွေက Software Engineering ပိုင်းတို့၊ Product Management ပိုင်းတို့ကို ရောက်လာတာပါ။ စနစ်တကျအချိန်ယူပြီး စာတွေ့လက်တွေ့ လေ့လာရတဲ့ အကြောင်းအရာတွေပါ။

AI က ကုဒ်တွေရေးပေးတယ်၊ အလုပ်လုပ်သွားလို့ Publish လုပ်ထားလိုက်တယ် ဆိုရုံ မဟုတ်ဘဲ အခုလိုကိစ္စလေးတွေပါ ထည့်သွင်းစဉ်းစားထားမှသာ လက်တွေ့အသုံးချလို့ရ တဲ့အဆင့်ကို ရောက်မှာပဲဖြစ်ပါတယ်။

နောက်ပိုင်းမှာ တကယ်လုပ်ကြည့်တဲ့အခါ တွေ့လာပါလိမ့်မယ်။ တချို့ဘယ်လိုမှ ညှိလို့မ ရတဲ့ အသေးအဖွဲ ပြဿနာလေးတွေ ရှိလာတယ်။ ညှိလိုက်တာနဲ့ တခြားမဆိုင်တဲ့ နေရာ တွေမှာ ပြဿနာတွေ တက်သွားတယ်။ တချို့ Error တွေက ဘယ်လိုမှကို ဖြေရှင်းလို့မရ ဘူး ဖြစ်လာတယ်။ နောက်ထပ် ဘာကိုမှ ထပ်ထည့်လို့ မရဖြစ်လာတယ်။ တခုခု ထပ် ထည့်လိုက်ရင် ရှေ့က အဆင်ပြေပြီးသားတွေ အဆင်မပြေဖြစ်သွားတယ်။ စသည်ဖြင့် ကြုံတွေ့လာရပါလိမ့်မယ်။

ပြဿနာတော်တော်များများက AI နဲ့ တိုင်ပင်ဖြေရှင်းလိုက်ရင် ရတတ်ပေမယ့်၊ အသုံးချ သူကိုယ်တိုင်က သက်ဆိုင်ရာ နည်းပညာတွေကို နားလည်မှသာ ဖြေရှင်းလို့ရမယ့်ကိစ္စတွေ ရှိလာပါလိမ့်မယ်။

ဒီစာအုပ်ရဲ့ ရည်ရွယ်ချက်က စာဖတ်သူကို AI အသုံးချပြီး App တွေ ရေးသားတတ်စေ ချင်ရုံတင်မကဘဲ၊ ဒီလိုသိရှိသင့်တဲ့ သဘောတရားတွေကိုလည်း သိရှိထားပြီး ဆက်လက် လေ့လာသွားစေချင်တာပဲ ဖြစ်ပါတယ်။

နောက်ဆက်တွဲ

အခုဆိုရင် ဒီစာအုပ်မှာဖော်ပြချင်တဲ့ အကြောင်းအရာတွေ စုံသလောက် ဖြစ်သွားပါပြီ။ ဒီအခန်းမှာတော့ ကုန်ကျစရိတ်တွေကို သက်သာစေပြီး အခမဲ့လည်းရနိုင်တဲ့ အရံ နည်းလမ်းတွေကို ဖြည့်စွက် ဖော်ပြသွားမှာပါ။

Open Models

DeepSeek လို့ခေါ်တဲ့ တရုတ်မော်ဒယ်တစ်ခုကနေ အစပြု ကိုင်လှုပ်လိုက်လို့ ကနေ့အချိန်မှာ ထိပ်တန်း AI တွေနဲ့ စွမ်းဆောင်ရည် မတိမ်းမယိမ်းရှိပေမဲ့ ကုန်ကျစရိတ်မှာ တော်တော်ကွာခြား သက်သာတဲ့ တရုတ်မော်ဒယ်တွေ ရှိနေကြတာ အားလုံးသိကြပါတယ်။

အဲ့ဒီ Model တွေအများစုက Open Model တွေ ဖြစ်ကြပါတယ်။ ကိုယ့်မှာသာ Run နိုင်တဲ့ စက်ရှိရင် အခမဲ့ Download ရယူပြီး Run လို့ရတဲ့အပြင်၊ သူဘယ်လို Train ထားလဲဆိုတဲ့ အချက်အလက်တွေကိုပါ ဖွင့်ချပေးထားတဲ့ Model တွေပါ။

အများစုက MoE လို့ခေါ်တဲ့ Mixture of Experts စနစ်ကို အသုံးပြုကြပါတယ်။ တခုခု အလုပ်လုပ်တော့မယ်ဆိုရင် Model ထဲက ရှိသမျှ Neuron တွေ အကုန် Run စရာ မလိုဘဲ သက်ဆိုင်ရာ Neuron တွေကိုပဲ ရွေး Run ပြီး အလုပ်လုပ်ပေးနိုင်လို့ ကွန်ပျူတာ စွမ်းဆောင်ရည် လိုအပ်ချက်နည်းနည်းနဲ့ အလုပ်တွေများများ လုပ်ပေးနိုင်ပါတယ်။

လိုအပ်ချက်နည်းတယ်ဆိုတာက ဒီနည်းကို သုံးမထားတဲ့ တခြား Model တွေနဲ့ယှဉ်ရင် နည်းတာပါ။ လိုအပ်ချက်နည်းတယ်ဆိုလို့ ကြိုက်တဲ့ကွန်ပျူတာမှာ ထည့် Run လို့ ရသွားပြီဆိုတဲ့ အဆင့်ထိတော့ မရောက်သေးပါဘူး။ ဘာပဲပြောပြော ဒီလိုစွမ်းဆောင်ရည် လိုအပ်ချက်နည်းလို့ သူတို့ရဲ့ API တွေကို ရယူအသုံးပြုမယ်ဆိုရင် တခြား ထိပ်တန်း Model တွေရဲ့ ဈေးနှုန်း အဖျားအနားလောက်နဲ့ တော်တော်လေး သက်သက်သာသာ အသုံးပြုလို့ရနိုင်မှာ ဖြစ်ပါတယ်။

Open Model ဆိုပြီး ဘာဖြစ်လို့ ကုန်ကျစရိတ် ရှိသလဲဆိုတော့ ကိုယ်စက်ထဲမှာ Run မရဘဲ သူတို့ဆာဗာတွေပေါ်မှာ Run ထားတာကို လှမ်းယူသုံးရလို့ အဲ့ဒီအတွက် အဖိုးအခကို ပေးရတဲ့သဘောပါ။

ဒီလို Open Model တွေထဲမှာ အရင်ကဆိုရင် Facebook ရဲ့ Llama Model တွေက အဓိကပါပဲ။ နောက်ပိုင်းမှာ Llama Model တွေက သတင်းခေါင်းစီး ဖြစ်မနေတော့ဘဲ

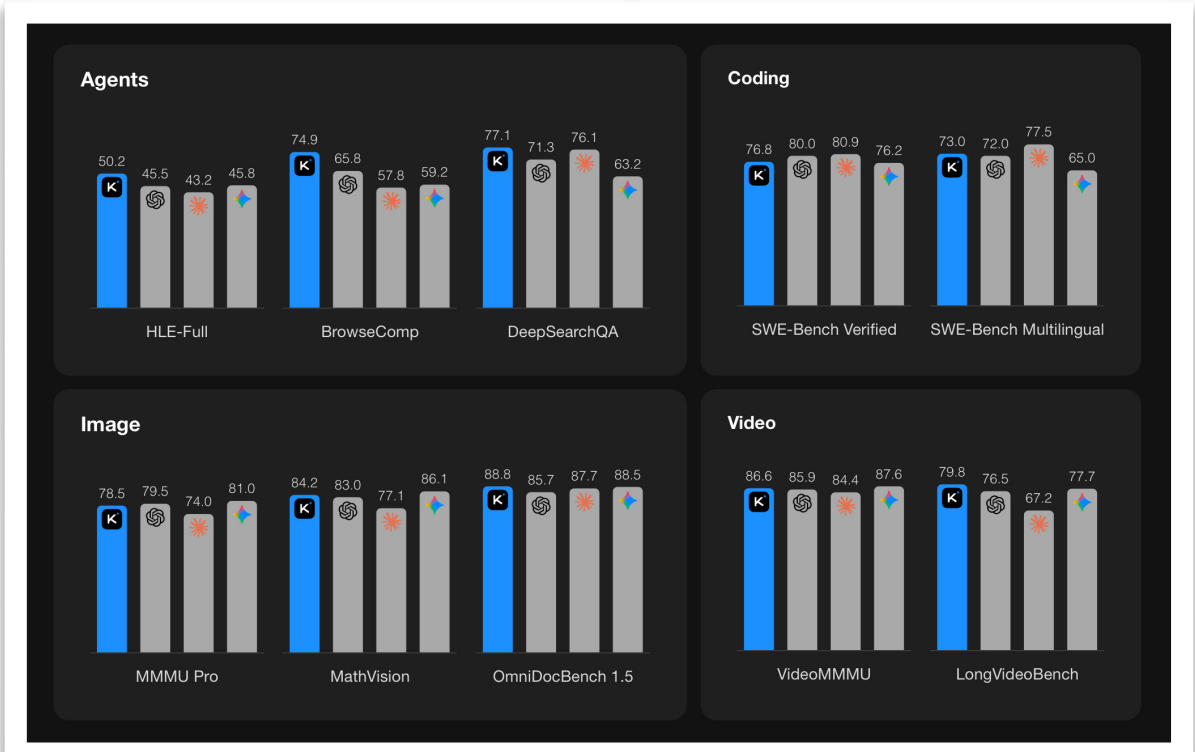
ဆက်တိုက်ဆိုသလို အသစ်ထွက်နေတဲ့ တရုတ် Model တွေက ပိုပြီးတော့ စိတ်ဝင်စားစရာ ဖြစ်လာနေတာပါ။

လက်ရှိ ဒီစာရေးသားနေစဉ် လူတွေ စိတ်ဝင်စားမှု အများဆုံး Open Model တွေကတော့ -

- Kimi K2.5
- GLM 4.7 နဲ့
- MiniMax M2.1

- တို့ ဖြစ်ကြပါတယ်။

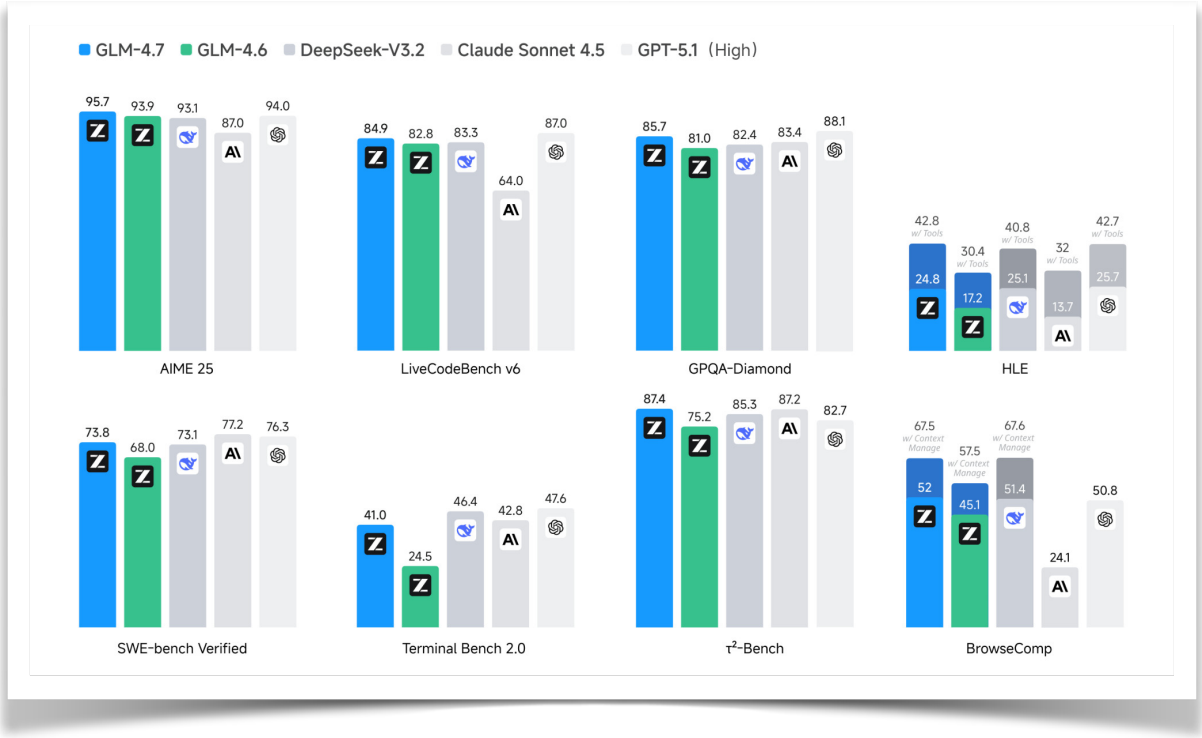
အောက်ကပုံမှာကြည့်လိုက်ရင် **Kimi K2.5** က GPT, Claude, Gemini တို့နဲ့ ရင်ဘောင်တန်းနေပြီး တချို့အပိုင်းလေးတွေမှာ အနည်းငယ်တောင် ပိုသာသေးနေသေးတယ်လို့ ပြထားတာကိုတွေ့ရနိုင်ပါတယ်။



Model အားလုံးလိုလိုက ဒီလို ကိုယ်ပိုင် Benchmark တွေမှာ ကိုယ့် Model က အသာရအောင် စမ်းသပ်ထုတ်ပြန်လေ့ ရှိကြတာကိုတော့ သတိပြုရပါမယ်။

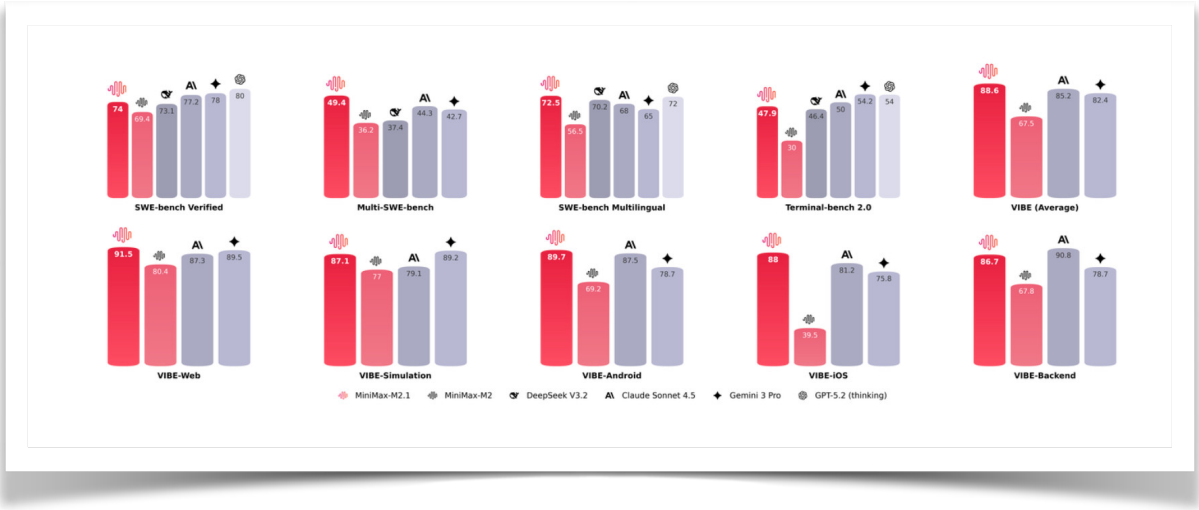
https://www.kimi.com

GLM 4.7 ကလည်းသူ့ရဲ့ Benchmark ကို အခုလိုပြထားပါတယ်။ ထိပ်တန်း Model တွေနဲ့ မတိမ်းမယိမ်းဖြစ်သလို တချို့နေရာတွေမှာ သူကတောင် ပိုသာနေတာပါ။



https://chat.z.ai/

MiniMax M2.1 ကလည်း အားတူပဲ တိုင်းတာပြထားတာကို တွေ့ရနိုင်ပါတယ်။



https://www.minimax.io/

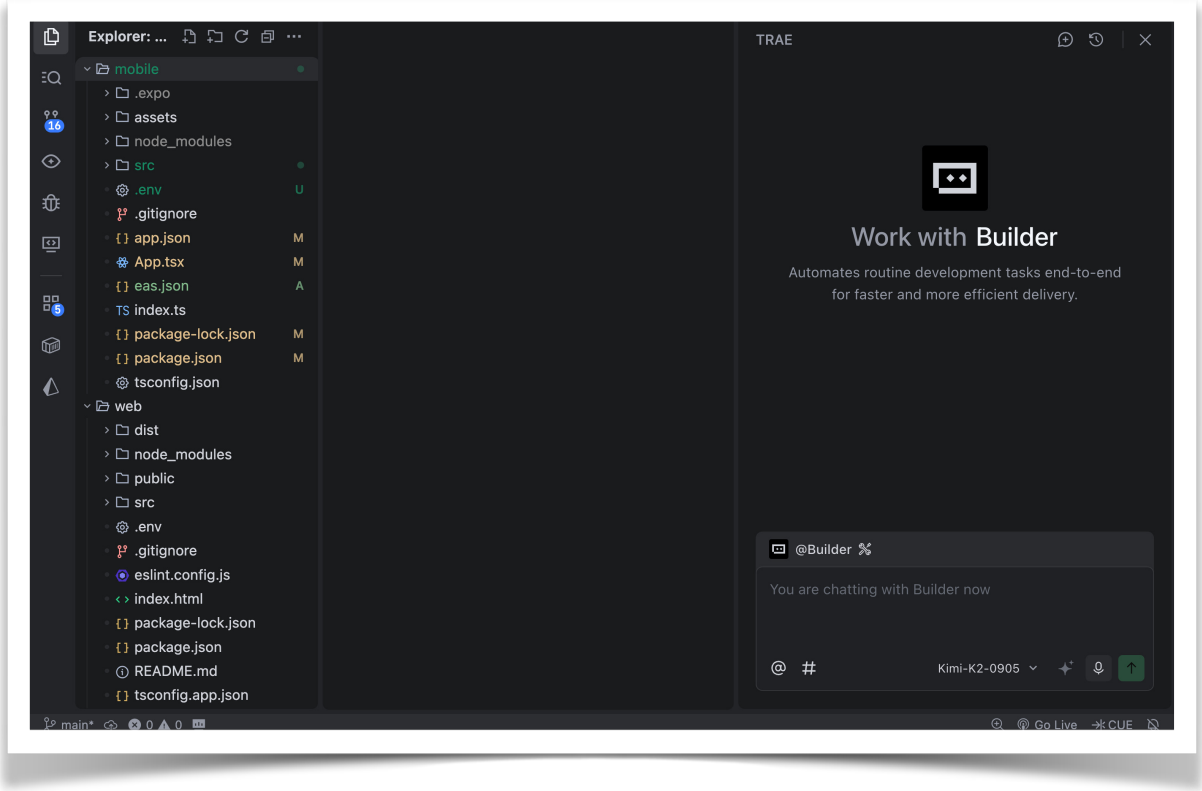
လောလောဆယ် ဒီသုံးခုကို အထူးသဖြင့် လူတွေစိတ်ဝင်စားနေပေမဲ့ ဒါတွေက ကြောင်နဲ့ ကြွက် လိုက်တမ်းပြေးတမ်း ကစားသလို၊ သူကသာသွားလိုက် ငါကသာသွားလိုက်နဲ့ အသစ်အသစ်တွေ အမြဲလိုလို ထွက်နေကြတာပါ။ စာဖတ်သူ ဒီစာကို ဖတ်နေချိန်ဆိုရင် နောက်ထပ် Version သစ်တွေတောင် ထပ်ထွက်နေလောက်ပါပြီ။

သူတို့နဲ့အတူ Vibe Coding အတွက် နောက်ထပ် ပူးတွဲပြီး စောင့်ကြည့်သင့်တာတွေ ကတော့ မူလလက်ဟောင်း DeepSeek နဲ့ Qwen လို့ခေါ်တဲ့ Model တွေဖြစ်ပါတယ်။

DeepSeek - https://www.deepseek.com/en/

Qwen - https://qwen.ai/

ဒီ Model တွေကို စမ်းသပ်အသုံးပြုဖို့အတွက် သင့်တော်တဲ့ နည်းပညာကတော့ Trae AI လို့ခေါ်တဲ့ Cursor လိုမျိုး Code Editor Agent ပါ။ အောက်ကလင့်မှာ Download ရယူနိုင်ပါတယ်။



https://www.trae.ai/

နမူနာပုံက Trae AI Agent မှာ Kimi K2 ကို ရွေးထားတာပါ။ K2.5 တော့ မဟုတ်သေးပေမဲ့ သူ့နေရာနဲ့သူ အသုံးဝင်ပါတယ်။

ဒီလိုမျိုး Open Model တွေနဲ့ ပူးတွဲအသုံးပြုဖို့ နောက်ထပ်စိတ်ဝင်စားဖို့ကောင်းတဲ့ နည်းပညာကတော့ **OpenCode** လို့ခေါ်ပါတယ်။ Claude Code လိုမျိုး CLI Agent ကိုမှ သူက

Open Source ဖြစ်သွားတာပါ။ လက်ရှိ ဒီစာရေးနေစဉ်မှာ လူတွေအရမ်းကြိုက်နေကြတဲ့ နည်းပညာပါ။ Claude Code ထက်တောင် ပိုကောင်းသေးတယ်လို့ ပြောတဲ့သူတွေက ပြောကြပါတယ်။

http://opencode.ai/

CLI Agent ဖြစ်တဲ့အတွက် Terminal မှာ အခုလို Install လုပ်လို့ရပါတယ်။

```
npm i -g opencode-ai
```

ပြီးတဲ့အခါ ပရောဂျက်ဖိုဒါထဲမှာ Terminal ဖွင့်ပြီး အခုလို သုံးလို့ရပါတယ်။

```
opencode
```

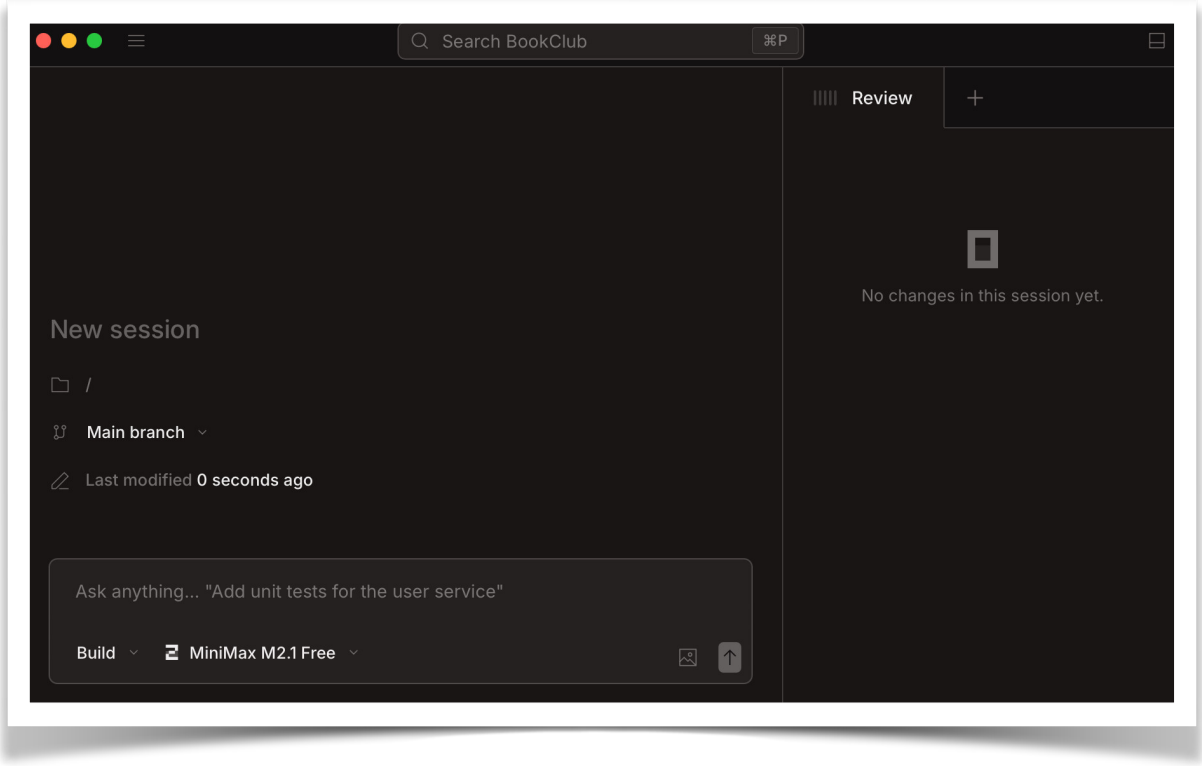
```
/models
```

လို့ ရိုက်ထည့်ကြည့်လိုက်ရင် အခုလို Kimi K 2.5 နဲ့ MiniMax M 2.1 တို့ကို Free အသုံးပြုခွင့်ပေးထားတာ တွေ့နိုင်ပါတယ်။

သူက CLI Agent ဆိုပေမဲ့ Desktop App လည်းပေးထားပါတယ်။ တကယ်ကောင်းတာ တော့ CLI Agent က ပိုကောင်းပေမဲ့ အသုံးပြုရပိုလွယ်တာကို သုံးချင်ရင် Desktop App ကို Download လုပ်ယူပြီး သုံးလို့လည်း ရပါတယ်။

အောက်ကလိပ်စာမှာပဲ Down လို့ရပါတယ်။

http://opencode.ai/



ပုံမှာတွေ့မြင်နေရတာကတော့ ပရောဂျက်ကို Open Code App နဲ့ ဖွင့်ထားတာဖြစ်ပြီး MiniMax M2.1 Free ကို ရွေးချယ်ထားပါတယ်။

ပုံမှန်အားဖြင့် Codex တို့ Claude တို့ Gemini တို့လို ထိပ်တန်း Model တွေနဲ့ အလုပ်လုပ်ပြီး Limit ပြည့်သွားလို့ပဲဖြစ်ဖြစ် အရေးပေါ်လိုအပ်လာတဲ့အခါ အခုလို အခမဲ့ (သို့) အခမဲ့နီးပါးရတဲ့ နည်းပညာတွေကို အစားထိုး အသုံးပြုလို့ရပါတယ်။

တစ်ခုပြီးတစ်ခု ပြောင်းသုံးရင် အလုပ်ရှုပ်ပေမဲ့ ကုန်သွားတယ်၊ မလောက်ဘူးဆိုတာ မရှိတော့ပါဘူး။ သူ့ဘာသာ လည်ပတ် အဆင်ပြေသွားပါတယ်။ ဒါက စာရေးသူ ကိုယ်တိုင် အသုံးပြုနေတဲ့ နည်းတွေပါ။

Local Model

နောက်ဆုံး “အရံ” နည်းလမ်းက စိတ်ဝင်စားဖို့ကောင်းပေမဲ့ သိပ်လက်တွေ့ မကျလို့ ထည့်မပြောတော့ဘူးလို့ နေတာပါ။ ဒါပေမဲ့ လိုရမယ်ရ ထည့်ပြောလိုက်ပါမယ်။

ကိုယ့်ရဲ့စက်ထဲမှာ Model ကို Download လုပ်ထည့်သွင်းပြီး Run တဲ့နည်းလမ်းပါ။ လက်တွေ့မကျဘူးဆိုတာက ဒီလိုပါ။ အဲ့ဒီလို Run ဖို့အတွက် တော်တော်ကောင်းတဲ့ စက်လိုပြီး၊ ဒေါ်လာသောင်းချီတန်အောင် တော်တော်ကောင်းတဲ့စက်ရှိပါတယ် ဆိုရင်တောင် အကောင်းဆုံးဆိုတဲ့ Model တွေကို Run လို့ရဦးမှာ မဟုတ်သေးပါဘူး။ အလယ်အလတ် အဆင့် Model တွေလောက်ပဲ Run လို့ရမှာပါ။

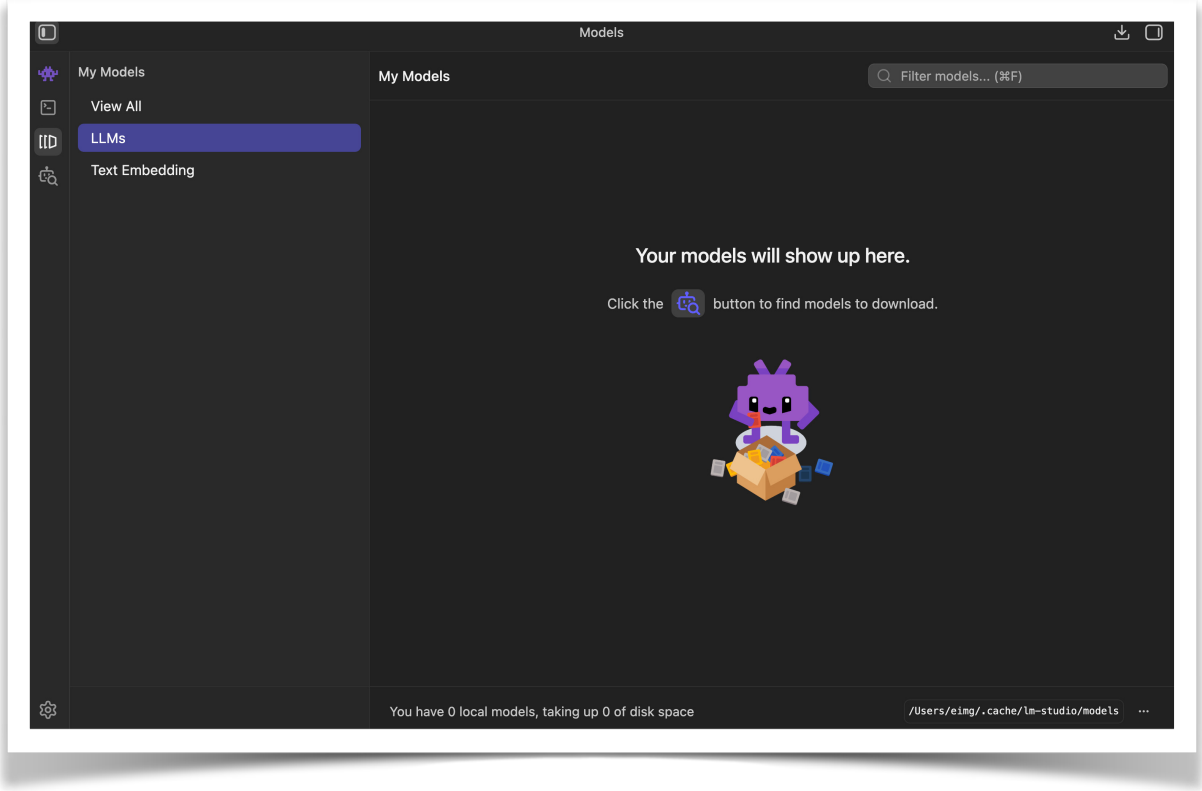
ဒါကြောင့် စိတ်ချလက်ချ သုံးရလောက်အောင် Smart မဖြစ်ကြဘဲ စမ်းကြည့်လို့ရတယ် ဆိုယုံလောက်ပဲ ရှိကြပါတယ်။

စာရေးသူလက်ရှိ အသုံးပြုနေတဲ့စက်က M3 MacBook Pro ကို 16 GB RAM နဲ့ပါ။ အညံ့ကြီးမဟုတ်သလို၊ အကောင်းကြီးလည်း မဟုတ်ပါဘူး။ အလယ်အလတ်အဆင့်တောင် Run လို့မရဘဲ Model ခပ်သေးသေးလေးတွေကိုသာ Run လို့ရပါတယ်။

ဘာပဲပြောပြော ဗဟုသုတရအောင် ကြည့်ထားချင်တယ်၊ စမ်းကြည့်ချင်တယ်ဆိုရင် ဘယ်လိုလုပ်ရလဲ သိရအောင် ထည့်ပြောလိုက်ပါမယ်။

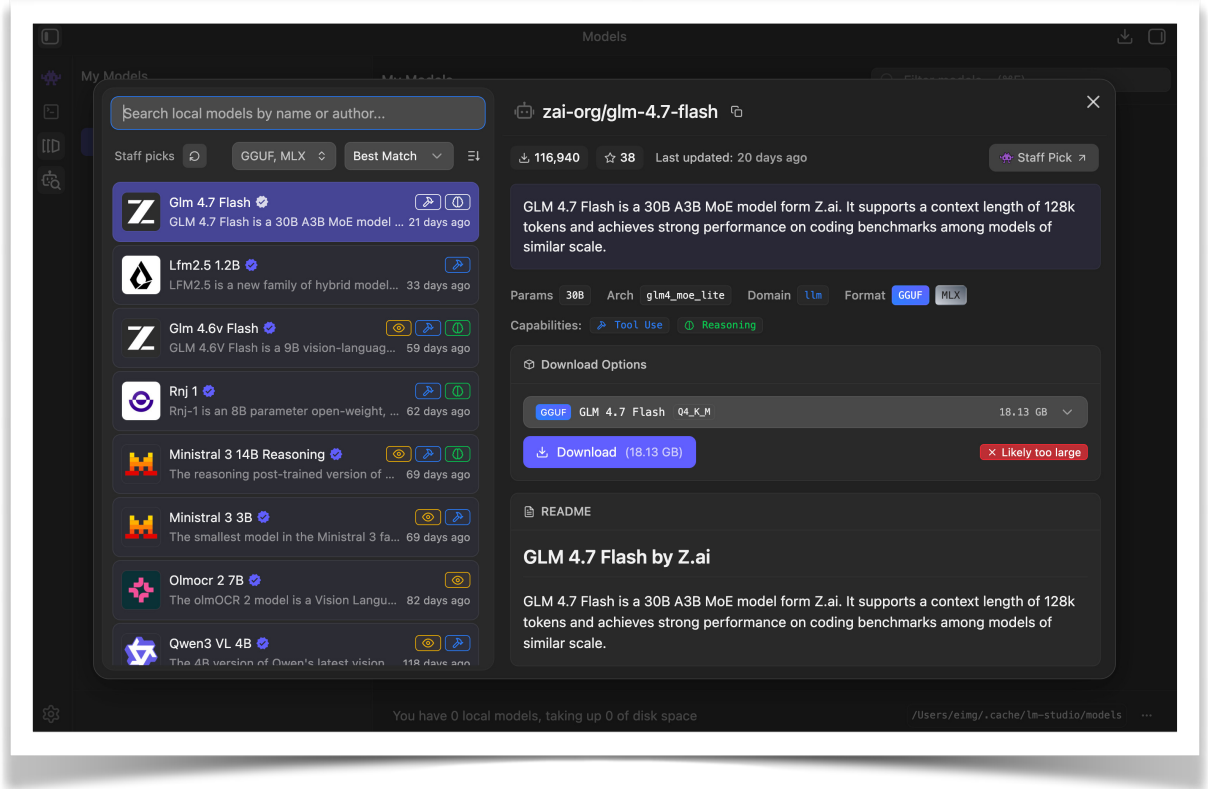
နည်းပညာတွေအများကြီး ရှိတဲ့ထဲက **LM Studio** လို့ခေါ်တဲ့ နည်းပညာကို အသုံးပြုမှာပါ။ ပေးထားတဲ့ လင့်ကနေ Download လုပ်ပြီး Install လုပ်လို့ရပါတယ်။

https://lmstudio.ai/



နမူနာမှာ တွေ့မြင်နေရတာက Install လုပ်ပြီးနောက် ရရှိမယ့် ရလဒ်ဖြစ်ပါတယ်။

ဘယ်ဘက်အစွန်ဆုံးက Icon ခလုတ်လေးတွေ နှိပ်ကြည့်လို့ရပါတယ်။ အောက်ဆုံးခလုတ်ကိုနှိပ်ပြီး လိုချင်တဲ့ Model တွေ ရှာဖွေ Download လုပ်ယူလို့ရပါတယ်။



သတိပြုစရာအချက်လေးတချို့ ရှိပါတယ်။ ပထမတစ်ခုက Model Format ပါ။ **GGUF** နဲ့ **MLX** ဆိုပြီး ရှိပါတယ်။ အကျဉ်းချုပ်အားဖြင့် MLX ဆိုတာ Mac Format ဖြစ်ပြီး GGUF ဆိုတာ Windows အပါအဝင် ပလက်ဖောင်းအမျိုးမျိုးနဲ့ သင့်တော်တဲ့ Format ပါ။ ML Studio က ကိုယ့်စက်နဲ့ သင့်တော်တာတွေကို ရွေးထုတ်ပြပေးလေ့ရှိပါတယ်။

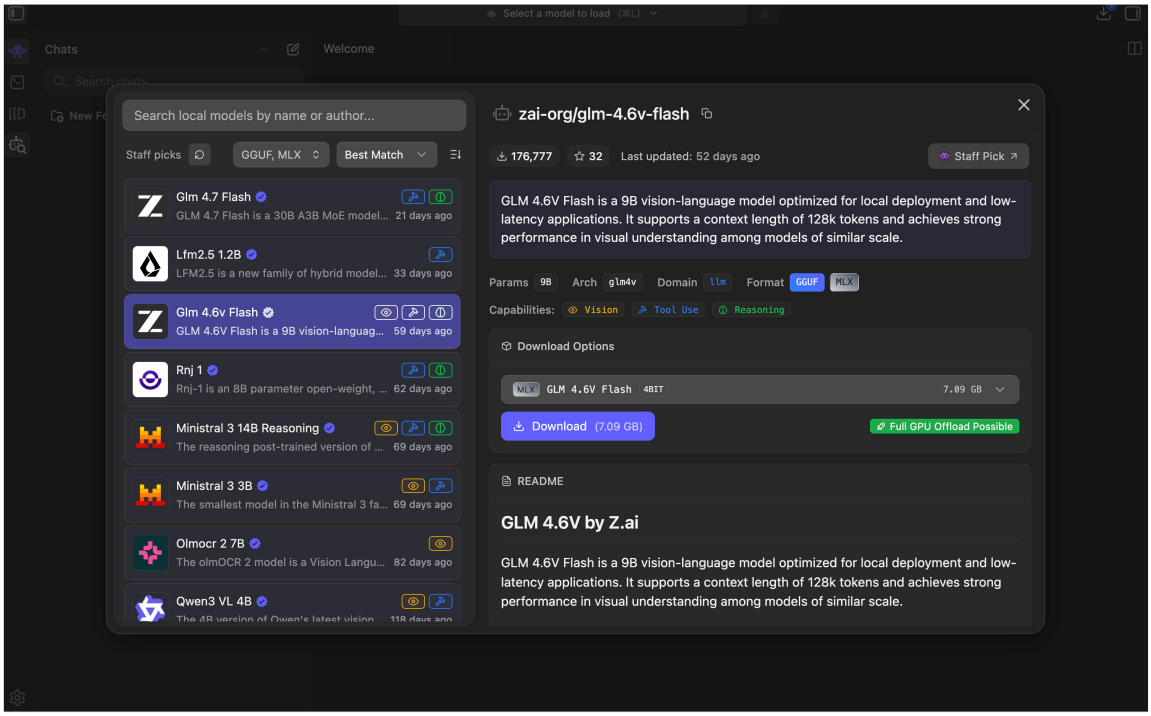
နမူနာပုံမှာကြည့်လိုက်ရင် သူရွေးထုတ်ပေးထားတဲ့ Model စာရင်းကိုတွေ့ရပြီး လက်ရှိ **GLM 4.7 Flash** လို့ခေါ်တဲ့ Model ကို ရွေးထားပါတယ်။ ညာဘက်ခြမ်းမှာ စာအနီနဲ့ ဒီစက်အတွက် ကြီးလွန်းတယ်လို့ ပြောထားတာကိုလည်း တွေ့နိုင်ပါတယ်။

စာရေးသူရဲ့စက်မှာ Run လို့မရလောက်ပါဘူး။ ဖိုင် Size 18.13 GB ရှိပါတယ်။ စာရေးသူရဲ့စက်က RAM 16 GB ပဲ ရှိပါတယ်။

Model တွေနဲ့ ပက်သက်ပြီး နောက်ထပ် သတိပြုစရာတစ်ခုရှိပါတယ်။ Parameter အရွယ်အစားပါ။ Model အမည်မှာ (သို့မဟုတ်) Description မှာ ပါနိုင်ပါတယ်။ နမူနာ အရ GLM 4.7 Flash Model ဟာ **30B** Model ဖြစ်ပါတယ်။ Parameter ပေါင်း (၃၀) ဘီလီယံနဲ့ Train ထားတာပါ။ အကျဉ်းချုပ်အားဖြင့် Parameter များလေ စွမ်းဆောင်ရည် ကောင်းလေပါပဲ။ Model အရွယ်အစားကြီးလေ၊ Run ဖို့ ပိုခက်လေပါပဲ။

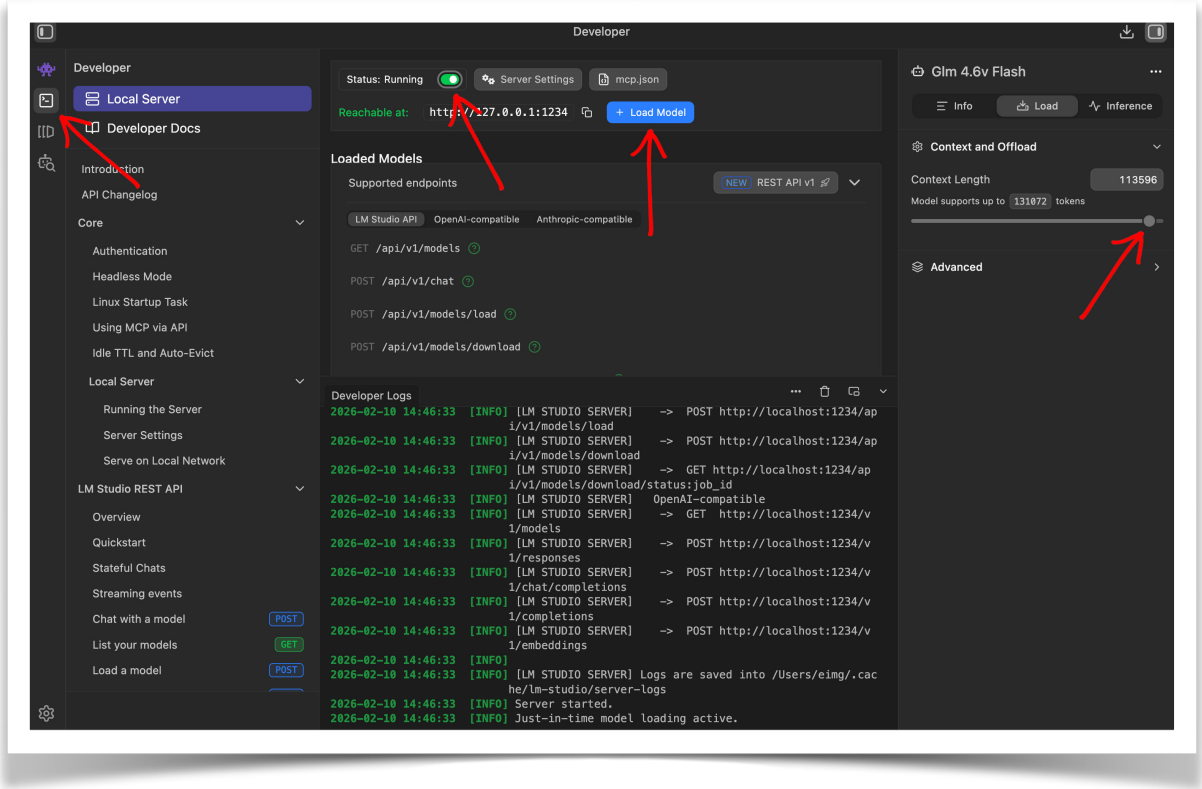
200B လောက်ထိ Run နိုင်ရင် မဆိုးပါဘူး။ စာရေးသူရဲ့စက်က **30B** တောင် Run လို့ အဆင်မပြေလောက်ပါဘူး။ တကယ့် ထိပ်တန်း Model တွေရဲ့ Parameter က အတိအကျမသိရပေမဲ့ **1T-2T** လောက်ထိ ဖြစ်နိုင်ပါတယ်။ တော်တော့်ကို ကွာပါတယ်။

လောလောဆယ် စမ်းကြည့်ဖို့အတွက် စက်နဲ့အဆင်ပြေလောက်မယ့် **GLM 4.6v Flash** (9B) Model ကို ဒေါင်းလိုက်ပါတယ်။ ဖိုင် Size က 7.09 GB ပဲ ရှိပါတယ်။



သူက Vision ပါတဲ့ Model ပါ။ Coding သီးသန့် Model မျိုးတော့ မဟုတ်ပါဘူး။ ဒါပေမဲ့ Tool Use တွေ Reasoning တွေရတဲ့အတွက် Coding အတွက်လည်း အဆင်ပြေလိမ့်မယ်လို့ ထင်ပါတယ်။

Download လုပ်ပြီးသွားပြီဆိုရင် အဲ့ဒီ Model နဲ့ သူ့မှာ အဆင်သင့်ပါတဲ့ Chat UI ကနေ ကုဒ်ရေးခိုင်းတာတွေ၊ အမေးအဖြေလုပ်တာတွေ လုပ်လို့ရပါပြီ။ ဒီမှာတော့ Coding Agent နဲ့ ချိတ်ချင်တာဖြစ်လို့ ဘယ်ဘက်ခြင်းက Developer Icon ခလုတ်လေးကို နှိပ်လိုက်ပါတယ်။



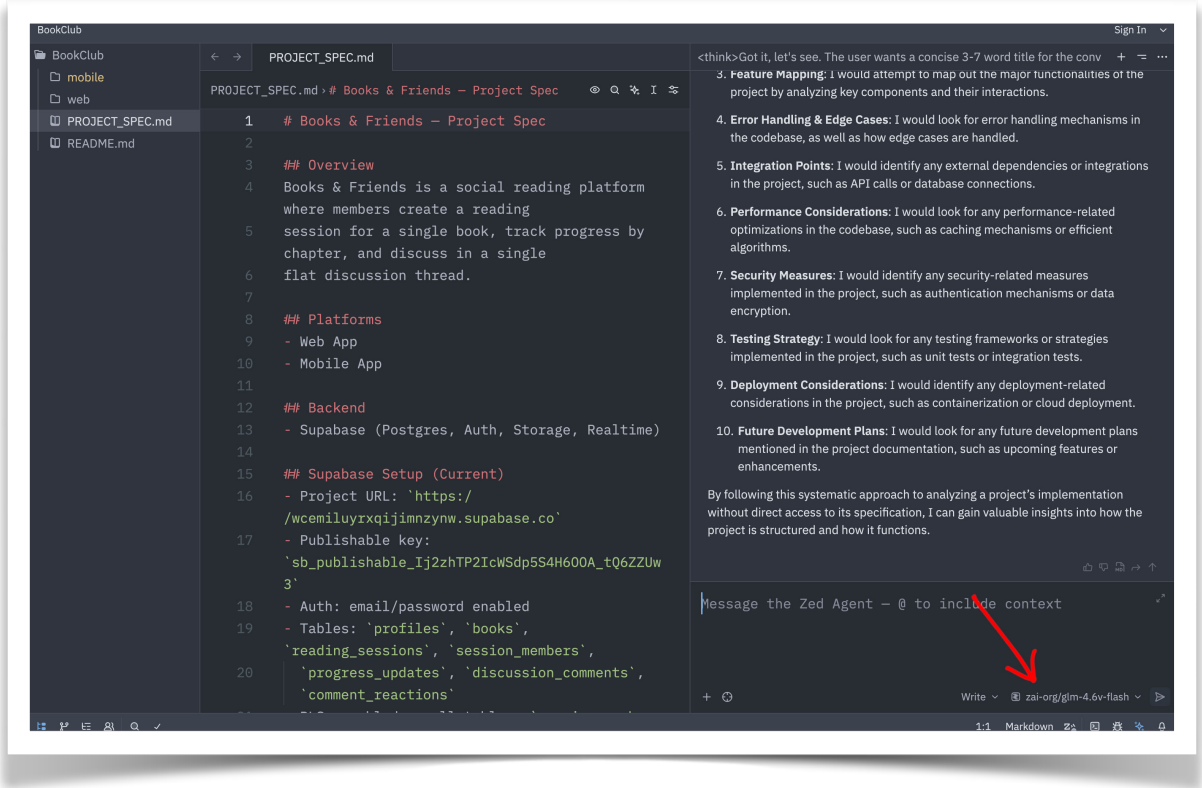
ပုံမှာ မြှားပြထားသလို Load Model နဲ့ Download လုပ်ထားတဲ့ Model ကို ရွေးလိုက်ပါ။ ပြီး ရင် Status Running ဖြစ်နေအောင် ရွေးထားပေးပါ။ ညာဘက်အစွန်ဆုံးမှာ ပြထား

တာက Context Length ကို ကိုယ့်စက်က လက်ခံနိုင်သလောက် များများ ပေးထားလိုက်တာပါ။

ဒါဟာ Download လုပ်ထားတဲ့ Model ကို LM Studio နဲ့ ဆာဗာတစ်ခုကဲ့သို့ Run ထားလိုက်တာပါ။ သူ့ကို **OpenCode** တို့ **Cursor** တို့နဲ့ ချိတ်ချင်ရင် ချိတ်လို့ရပါတယ်။ ပို လွယ်ချင်ရင်တော့ **Zed** လို Agent ကို အသုံးပြုနိုင်ပါတယ်။

https://zed.dev/

Zed ဟာ Cursor လို Code Editor Agent တမျိုးပါပဲ။



ပုံမှာပြထားသလို ချိတ်ပေးနေစရာတောင်မလိုဘဲ ဖွင့်လိုက်တာနဲ့ ချိတ်မိပြီး အဆင်သင့်တန်းသုံးလို့ ရနေတယ်ဆိုတာကို တွေ့ရပါလိမ့်မယ်။

ဒါဟာ ကိုယ့်စက်ထဲမှာ Run နေတဲ့ Local Model ကို အသုံးပြုပြီး ကုဒ်ရေးသားလိုက်တာပါပဲ။ စွမ်းဆောင်ရည်အကန့်အသတ်ရှိပေမဲ့ တော်တော် ပျော်ဖို့ကောင်းပါတယ်။ ကုန်ကျစရိတ်တွေလည်း မရှိတော့ဘူး။ ပြင်ပ Model API တွေကို လှမ်းချိတ်နေစရာလည်း မလိုတော့ဘူး။ ကိုယ့်စက်ထဲမှာတင် အားလုံးရသွားတာပါ။

တကယ်အမြဲတမ်း မသုံးဘူးဆိုရင်တောင် ဗဟုသုတရအောင် စမ်းကြည့်သင့်ပါတယ်လို့ တိုက်တွန်းပါတယ်။

နိဂုံးချုပ်

ဒီစာအုပ်မှာဖော်ပြချင်တဲ့ အကြောင်းအရာတွေ ပြည့်စုံ ပြီးဆုံးသွားပြီ ဖြစ်ပါတယ်။ စာဖတ်သူအတွက် အသုံးဝင်တဲ့ စာအုပ်တစ်အုပ် ဖြစ်လိမ့်မယ်လို့ မျှော်လင့်ပါတယ်။

ဒီစာအုပ်က Baseline အခြေတစ်ခုကို ရှင်းအောင် ချပေးလိုက်နိုင်လိမ့်မယ်လို့လည်း ထင်ပါတယ်။ Coding နည်းပညာတွေ နားလည်သည်ဖြစ်စေ၊ နားမလည်သည်ဖြစ်စေ၊ ဒါဟာ အားလုံးအတွက် Baseline ပါပဲ။

ဒီစာအုပ်မှာ ဖော်ပြထားတဲ့နည်းတွေနဲ့ လူတိုင်း လူတိုင်း App တွေဖန်တီးလို့ရပါတယ်။ လွှင့်တင်ဖြန့်ဝေနိုင်ပါတယ်။

Coding နည်းပညာ ကျွမ်းကျင်သူများက အဲ့ဒီ Baseline ပေါ်ကနေ နောက်အဆင့်တွေကို ထပ်ဆင့်တက်လှမ်းကြရမှာ ဖြစ်ပြီး၊ Coding နည်းပညာ မသိသူများက၊ သိချင်ရင် ဘာတွေဆက်လုပ်ရမယ်ဆိုတာကို ရှင်းသွားလိမ့်မယ်လို့ ယူဆပါတယ်။

ဆက်လုပ်သင့်၊ မလုပ်သင့်၊ ဆက်လုပ်လို့ ဖြစ်နိုင်၊ မဖြစ်နိုင်၊ ဒါတွေလည်း ကိုယ်တိုင် ဆုံးဖြတ်နိုင်သွားလိမ့်မယ်လို့ ယူဆပါတယ်။

အားလုံးပဲ အပြောင်းအလဲ စိန်ခေါ်မှုတွေကြောင့် အခက်မတွေ့ဘဲ၊ အလိုက်သင့် လိုက်ပါစီးမျော ခရီးနှင်နိုင်သူတွေ ဖြစ်ကြပါစေလို့ ဆုတောင်းပါတယ်။

အိမောင်

Fairway

(၂၀၂၆) ခုနှစ်၊ ဖေဖော်ဝါရီ (၁၂) ရက်နေ့တွင်
ရေးသားပြီးစီးသည်။