សាកលវិទ្យាល័យភូមិន្ទនីតិសាស្ត្រ និង វិទ្យាសាស្ត្រសេដ្ឋកិច្ច

Royal University of Law and Economics

សារណាបញ្ចប់ការសិក្សា

ការទិញទំនិញតាមប្រព័ន្ធអ៊ីនធឺណែត Online Shopping

ស្រាវជ្រាវចាប់ពីថ្ងៃទី ០៨ ខែមេសា ឆ្នាំ ២០២៤ ដល់ថ្ងៃទី ១០ សីហា ឆ្នាំ ២០២៤

ស្រាវជ្រាវដោយ
និស្សិតឈ្មោះ **លោក ស្រ៊ុន រដ្ឋា**

ណែនាំដោយ
សាស្ត្រាចារ្យជំនួយ **ឥន ហេង**

ថ្នាក់បរិញ្ញាបត្រ **ព័ត៌មានវិទ្យា**
ជំនាន់ទី ០៣

ឆ្នាំចូលសិក្សា ២០២០
ឆ្នាំសរសេរសារណា ២០២៤

សាកលវិទ្យាល័យភូមិន្ទនីតិសាស្ត្រ និង វិទ្យាសាស្ត្រសេដ្ឋកិច្ច

Royal University of Law and Economics

សារណាបញ្ចប់ការសិក្សា

ការធ្វើកិច្ចសន្យាតាមប្រព័ន្ធអ៊ីនធឺណែត Online Shopping

ស្រាវជ្រាវចាប់ពីថ្ងៃទី ០៨ ខែមេសា ឆ្នាំ ២០២៤ ដល់ថ្ងៃទី ១០ សីហា ឆ្នាំ ២០២៤

ស្រាវជ្រាវដោយ
និស្សិតឈ្មោះ **លោក ស្រ៊ុន រដ្ឋា**

ណែនាំដោយ
សាស្ត្រាចារ្យជំនួយ **ឥន ហេង**

ថ្នាក់បរិញ្ញាបត្រ **ព័ត៌មានវិទ្យា**
ជំនាន់ទី ០៣

ឆ្នាំចូលសិក្សា ២០២០
ឆ្នាំសរសេរសារណា ២០២៤

សេចក្តីថ្លែងអំណរគុណ

កិច្ចការសិក្សាស្រាវជ្រាវនេះបានធ្វើឡើងដោយនិស្សិតសាកលវិទ្យាល័យភូមិន្ទនីតិសាស្ត្រ និងវិទ្យាសាស្ត្រ សេដ្ឋកិច្ចជំនាន់ទី០៣ នៃមហាវិទ្យាល័យសេដ្ឋកិច្ចព័ត៌មានវិទ្យា ផ្នែកព័ត៌មានវិទ្យា ដែលមានឈ្មោះ **ស្រ៊ុន រដ្ឋា** ៖

សូមគោរពចំពោះ

លោកមាតាបិតា និង អាណាព្យាបាល របស់ខ្ញុំបាទ ដែលបានជួយទំនុកបម្រុង ដល់ការសិក្សារបស់ខ្ញុំ បាទតាំងពីថ្នាក់ បឋមសិក្សារហូតមកដល់ ថ្នាក់បរិញ្ញាបត្រមួយនេះ ដោយរូបលោកទាំងអស់ធ្វើការព្យាយាមធ្វើ ការទាំងថ្ងៃក្តៅ ដើម្បីបានថវិការដើម្បីមក ទំនុកបម្រុងដល់កូនៗ ដែលទង្វើទាំងអស់នេះ គឺជាទង្វើដែលមិនអាច កាត់ថ្លៃបាន ។

សូមថ្លែងអំណរគុណយ៉ាងជ្រាលជ្រៅចំពោះការទទួលស្វាគមន៍ និង ការជួយផ្តល់ខាងផ្នែកគំនិតស្មារតី របស់ ៖

- សាស្ត្រាចារ្យជំនួយ **ឆិន ហោច** នៃសាកលវិទ្យាល័យភូមិន្ទនីតិសាស្ត្រ និង វិទ្យាសាស្ត្រសេដ្ឋកិច្ចដែល បានផ្តល់ចំណេះដឹង ពេលវេលាដ៏មានតម្លៃ ក្នុងការជួយផ្តល់នូវគំនិតយោបល់ និងគន្លឹះល្អៗ ដើម្បីធ្វើការសិក្សា ស្រាវជ្រាវបន្ថែម ។

- សាស្ត្រាចារ្យទាំងអស់ចាប់តាំងពី ចូលមកសិក្សា ក្នុងសកលវិទ្យាល័យ ៤ឆ្នាំ មកនេះដែលបានជួយ បង្ហាត់បង្រៀនទាំងទ្រឹស្តី ការអនុវត្ត និងណែនាំនូវបទពិសោធន៍មួយចំនួនផងដែរ ។

ជាទីបញ្ចប់ ខ្ញុំបាទសូមប្រសិទ្ធពរជ័យ ដល់ លោកសាស្ត្រាចារ្យទាំងអស់ ព្រមទាំងលោកនាយក និង បុគ្គលិកទាំងអស់ នៃសាកលវិទ្យាល័យភូមិន្ទនីតិសាស្ត្រ និងវិទ្យាសាស្ត្រសេដ្ឋកិច្ច សូមឲ្យជួបតែពុទ្ធពរទាំង ៤ ប្រការ គឺ អាយុ វណ្ណៈ សុខៈ ពលៈ កុំបីឃ្លាឃ្លាតឡើយ។

អារម្ភកថា

ភាពជឿនលឿននៃ បច្ចេកវិទ្យាបានផ្លាស់ប្តូរ វិធីដែលយើងប្រាស្រ័យទាក់ទង ជាមួយពិភពលោកជុំវិញយើង។ ការផ្លាស់ប្តូរពីការលក់ បែបប្រពៃណីទៅជាវេទិកាទិញទំនិញអនឡាញ បានផ្លាស់ប្តូរអាកប្បកិរិយា ដោយងាកមក ប្រើប្រាស់បច្ចេកវិទ្យាគេហទំព័រ (Website Technology)វិញ ដើម្បីដោះស្រាយបញ្ហា និងងាយស្រួលប្រើ។ ការលើកទឹកចិត្តសម្រាប់គម្រោងនេះកើតចេញពីការកើនឡើង នៃតម្រូវការទិញទំនិញតាមប្រព័ន្ធអនឡាញប្រកបដោយប្រសិទ្ធភាព និងគួរឱ្យទុកចិត្ត។ គម្រោងនេះប្រើប្រាស់ បច្ចេកវិទ្យាមួយចំនួន រួមមាន Node.js, Express.js, MongoDB, Mongoose និង Reactjs ដើម្បីផ្តល់នូវបទពិសោធន៍ទិញទំនិញប្រកបដោយប្រសិទ្ធភាព។

ខ្ញុំសូមថ្លែងអំណរគុណយ៉ាងជ្រាលជ្រៅចំពោះ សាស្ត្រាចារ្យ របស់ខ្ញុំចំពោះការណែនាំ និងការគាំទ្រដ៏មានតម្លៃរបស់លោកក្នុងគម្រោងនេះ។ ការយល់ដឹង និងការលើកទឹកចិត្តរបស់គាត់ បានរួមចំណែកយ៉ាងសំខាន់ក្នុងការនាំយកគម្រោងនេះឱ្យទទួលបានផ្លែផ្កា។

គម្រោងនេះមិនត្រឹមតែបានពង្រឹងជំនាញបច្ចេកទេសរបស់ខ្ញុំប៉ុណ្ណោះទេ ប៉ុន្តែថែមទាំងធ្វើឱ្យការយល់ដឹងរបស់ខ្ញុំកាន់តែស៊ីជម្រៅអំពីភាពស្មុគស្មាញនៃការអភិវឌ្ឍន៍គេហទំព័រ។ ខ្ញុំសង្ឃឹមថា និក្ខេបបទនេះ និង បម្រើជាធនធានដ៏មានតម្លៃសម្រាប់មិត្តនិស្សិត អ្នកអភិវឌ្ឍន៍ (Developer) និង អ្នកដែលចាប់អារម្មណ៍លើវិស័យពាណិជ្ជកម្មអេឡិចត្រូនិក និងបច្ចេកវិទ្យាគេហទំព័រ (Website Technology) ។

សរុបសេចក្តីមក ខ្ញុំពិតជារំភើបក្នុងការបង្ហាញការងារនេះជា ចំណុចនៃដំណើរការសិក្សារបស់ខ្ញុំ ហើយខ្ញុំទន្ទឹងរង់ចាំក្នុងការរួមចំណែកដល់ ការវិវត្តន៍នៃបច្ចេកវិទ្យានាពេលអនាគត។

មាតិកា

ទំព័រ

ជំពូកទី១

ស្ថាបត្យកម្មប្រព័ន្ធ និងលំនាំរចនា (System Architecture and Design Pattern)

ជំពូកទី២

ការរចនាប្រព័ន្ធ (System Design)

ជំពូកទី៣

ការពិចារណាអំពីសុវត្ថិភាព (Security Consideration)

ជំពូកទី៤

បទពិសោធន៍អ្នកប្រើប្រាស់ និងការរចនា (User Experience and Design)

ជំពូកទី៥

លទ្ធផលនៃការស្រាវជ្រាវ

សេចក្តីសន្និដ្ឋាន និងការផ្តល់អនុសាសន៍

បញ្ជីរូបភាព

សេចក្តីផ្តើម

ការរីកចម្រើនយ៉ាងឆាប់រហ័សនៃពាណិជ្ជកម្មអេឡិចត្រូនិកបានផ្លាស់ប្តូររបៀបដែលអាជីវកម្មដំណើរការ និងរបៀបដែលអ្នកប្រើប្រាស់ធ្វើអន្តរកម្ម(**Interact**) ជាមួយផលិតផល (**Product**) និង សេវាកម្ម(**Service**)។ ការទិញតាំ់អនឡាញ ផ្តល់នូវភាពងាយស្រួល ភាពសម្បូរបែប និងតម្លៃល្អសមរម្យ។ ជាលទ្ធផល អាជីវកម្មជាច្រើននៅក្នុង និងក្រៅប្រទេសកំពុងងាកទៅរកវេទិកាអនឡាញ (**online shopping platform**) កាន់តែខ្លាំងឡើង ដើម្បីជំរុញការលក់។

ក្នុងយុគសម័យឌីជីថលនេះ ការកសាងហាងអនឡាញដ៏រឹងមាំ និងអាចរក្សាការប្រកួតប្រជែង។ ជម្រើសនៃបច្ចេកវិទ្យា ដើរតួនាទីយ៉ាងសំខាន់ក្នុងការកំណត់ភាពជោគជ័យ នៃវេទិកាទិញទំនិញអនឡាញ។ សារណានេះ នឹងផ្តោតលើការអភិវឌ្ឍន៍ហាងអនឡាញដោយប្រើបច្ចេកវិទ្យាទំនើប រួមមាន React.js សម្រាប់ផ្នែកខាងមុខ(Front-End) Node.js និង Express.js សម្រាប់ផ្នែកខាងក្រោយ(Back-End) និង MongoDB រួមជាមួយនឹង Mongoose សម្រាប់ការគ្រប់គ្រងមូលដ្ឋានទិន្នន័យ(Database)។

១.១. ចំនាំបញ្ហានៃការស្រាវជ្រាវ

ការវិវត្តន៍នៃបច្ចេកវិទ្យាបានផ្លាស់ប្តូរ ជាពិសេសនៅក្នុងវិស័យលក់រាយ។ E-commerce បានផ្តល់នូវភាពងាយស្រួលដែលមិនអាចប្រៀបផ្ទឹមបាន ភាពសម្បូរបែបនៃផលិតផល និងតម្លៃប្រកួតប្រជែង។ ដោយសារការលក់រាយបែបប្រពៃណីបន្តប្រឈមមុខនឹងបញ្ហា វេទិកាទិញទំនិញអនឡាញបានលេចចេញជាបណ្តាញដ៏សំខាន់សម្រាប់អាជីវកម្ម ដើម្បីទៅដល់អតិថិជន។ ការផ្លាស់ប្តូរនេះ ផ្តល់នូវបទពិសោធន៍អោយអ្នកប្រើប្រាស់ប្រកបដោយប្រសិទ្ធភាព។ ទោះជាយ៉ាងណាក៏ដោយ វាទាមទារអោយការរចនាផ្នែកខាងមុខ(Front-End) ដំណើរការផ្នែកខាងក្រោយ(Back-End) និងការគ្រប់គ្រងមូលដ្ឋានទិន្នន័យ(Data-Base)ធ្វើការជាមួយគ្នា ដើម្បីធានាបាននូវការប្រើប្រាស់ដោយមានភាពរលូន ចាប់ពីការស្វែងរកផលិតផល រហូតដល់ការបញ្ចប់ការទិញ។

បច្ចេកវិទ្យាដែលត្រូវបានជ្រើសរើសសម្រាប់ការបង្កើត **Online Shop** នេះមានដូចជា៖ **React.js**, **Node.js, Express.js, MongoDB,** និង **Mongoose** ផ្តល់មុខងារសំខាន់ៗជាច្រើនដែលធ្វើឱ្យវាល្អសម្រាប់បង្កើតហាងអនឡាញ (Online Shop) ។ **React.js** ផ្តល់អោយអ្នកងាយស្រួលប្រើប្រាស់ និង អនុញ្ញាតឱ្យមានការអាប់ដេតតាមពេលវេលាជាក់ស្តែង ។ **React.js** ផ្តល់នូវ component-based ដែលអាចអោយអ្នកបង្កើត (Developer)អាចបង្កើត Reusable UI ដែលអាចប្រើឡើងវិញបាន ដើម្បីអោយការអភិវឌ្ឍន៍កាន់តែមានប្រសិទ្ធភាព។ Node.js និង Express.js ប្រើរួមគ្នា គឺដើម្បីធ្វើការទៅលើដំណើរការអាជីវកម្ម **(core business**

logic)ដើម្បីធ្វើការតាមអ្វីដែលអ្នកប្រើប្រាស់ត្រូវការ ហើយមួយវិញទៀតប្រើ Node.js និងExpress.js គឺប្រើដើម្បីគ្រប់គ្រងទិន្នន័យ និងការទំនាក់ទំនងទៅកាន់ Back-End តាមរយៈ Mongoose ។ បច្ចេកវិទ្យាទាំងនេះនឹងរួមបញ្ចូលគ្នា ដើម្បីធ្វើការបង្កើត Online Shop នៃការសរសេរសារណានេះ ។

ដំណោះស្រាយសម្រាប់ការបង្កើត ហាងអនឡាញជាមួយនឹងបច្ចេកវិជ្ជាដែលបានជ្រើសរើស ដំណើរការអភិវឌ្ឍន៍ចាប់ផ្តើមជាមួយនឹង ផ្នែកខាងមុខ(Front-End) ផ្នែកខាងក្រោយ(**Back-End**) និងមូលដ្ឋានទិន្នន័យ (**DataBase**) ដើម្បីអាចប្រើប្រាស់បាន។ ផ្នែកខាងមុខ(**Front-End**) ដែលត្រូវបានបង្កើតឡើងដោយប្រើប្រាស់ React.js ផ្នែកខាងក្រោយជាមួយនឹង Node.js និង Express.js ដើម្បីធ្វើការគ្រប់គ្រងទំនាក់ទំនងជាមួយ DataBase ។ MongoDB ដែលគ្រប់គ្រងតាមរយៈ Mongoose ដើរតួជាDataBase ដើម្បីរក្សាទុកព័ត៌មាន និងកំណត់ត្រាប្រតិបត្តិការ។

១.២. ចំណោទបញ្ហានៃការដោះស្រាយ

ខណៈពេលដែលអាជីវកម្មជាច្រើនមានបំណងចង់បង្កើនការលក់លើអ៊ីនធឺណិត ពួកគេតែងតែប្រឈមមុខនឹងបញ្ហាទាក់ទងនឹងភាពស្មុគស្មាញនៃការអភិវឌ្ឍន៍គេហទំព័រ និងសមត្ថភាពក្នុងការគ្រប់គ្រងទិន្នន័យដ៏ច្រើនប្រកបដោយប្រសិទ្ធភាព។ ជាងនេះទៅទៀត ការរួមបញ្ចូលបច្ចេកវិទ្យាផ្សេងៗគ្នា អាចបង្កឧបសគ្គយ៉ាងសំខាន់សម្រាប់អ្នកអភិវឌ្ឍន៍ ជាពិសេសនៅពេលនិយាយអំពីការធានា ឱ្យមានការទំនាក់ទំនងរលូនរវាង **ផ្នែកខាងមុខ(Front-End)** **ផ្នែកខាងក្រោយ(Back-End)** ក៏ដូចជាការរក្សាភាពស៊ីសង្វាក់គ្នានៃទិន្នន័យ (**data consistency**)។ តើការបញ្ចូលបច្ចេកវិទ្យាចូលគ្នា ដើម្បីបង្កើតគេហទំព័រ (**Website**) ពិសេស React.js សម្រាប់ការអភិវឌ្ឍន៍ផ្នែកខាងមុខ(**Front-End**) , Node.js និង Express.js សម្រាប់ដំណើរការផ្នែកខាងក្រោយ (**Back-End**) និង MongoDB ជាមួយ Mongoose សម្រាប់ការគ្រប់គ្រងមូលដ្ឋានទិន្នន័យ (**DataBase**) - ដោះស្រាយបញ្ហាប្រឈមសំខាន់ៗដែលប្រឈមមុខ ដល់ម្ចាស់អាជីវកម្ម និងអ្នកប្រើប្រាស់បានអ្វីខ្លះបើប្រៀបធៀបទៅនឹងការលក់ ឬទិញទំនិញតាមបែបប្រពៃណី?

១.៣. គោលបំណងនៃការស្រាវជ្រាវ

គោលបំណងចម្បងនៃការស្រាវជ្រាវនេះគឺដើម្បីបង្កើតវេទិកាទិញទំនិញអនឡាញដ៏រឹងមាំ និងអាចធ្វើបានដោយប្រើបច្ចេកវិទ្យាដូចជា React.js, Node.js, Express.js និង MongoDB ។ តាមរយៈការរចនាប្រព័ន្ធតាមការស្រាវជ្រាវនេះរួមចំណែកធ្វើឱ្យការទិញទំនិញអនឡាញកាន់តែ មានប្រសិទ្ធភាព។ វាគាំទ្រសហគ្រាសធុនតូច និងមធ្យម (SMEs) ដោយផ្តល់នូវដំណោះស្រាយប្រកបដោយប្រសិទ្ធភាព ដែលអាចឱ្យពួកគេពង្រីក

ទីផ្សាររបស់ពួកគេដោយមិនចាំបាច់មានការវិនិយោគច្រើននោះទេ។ លើសពីនេះ ការស្រាវជ្រាវជំរុញឱ្យមានការ ជឿនលឿនផ្នែកបច្ចេកវិទ្យា ដោយបង្ហាញពីការប្រើប្រាស់ឧបករណ៍ទំនើបៗប្រកបដោយប្រសិទ្ធ។ ទីបំផុតនៃការ ស្រាវជ្រាវនេះមិនត្រឹមតែដោះស្រាយបញ្ហាបច្ចេកទេសប៉ុណ្ណោះទេ ប៉ុន្តែក៏ផ្តល់ប្រយោជន៍ ដល់សង្គមផងដែរ តាម រយៈការជំរុញការអនុវត្តពាណិជ្ជកម្មអេឡិចត្រូនិក គាំទ្រដល់កំណើនអាជីវកម្ម និងការជំរុញការបង្កើតថ្មីក្នុងការ អភិវឌ្ឍន៍គេហទំព័រ (Website)។

១.៤. ទំហំ និង ដែនកំណត់នៃការស្រាវជ្រាវ

សារណានេះផ្តោតលើទិដ្ឋភាពបច្ចេកទេសនៃការអភិវឌ្ឍន៍ហាងអនឡាញ (Online Shop)។ វិសាលភាពរួមបញ្ចូលការរចនាគេហទំព័រ និងការអនុវត្តទាំងផ្នែកខាងមុខ(Front-End) និងផ្នែកខាងក្រោយ(Back-End)ក៏ដូចជាការរួមបញ្ចូលប្រព័ន្ធទិន្នន័យ។ការសិក្សានេះមិនគ្របដណ្តប់លើយុទ្ធសាស្ត្រអាជីវកម្ម (Marketing Strategy) ដែលទាក់ទង នឹងពាណិជ្ជកម្មអេឡិចត្រូនិក ដូចជាទីផ្សារ ឬការការទាក់ទាញចិត្តអតិថិជននោះទេ ប៉ុន្តែសង្កត់ធ្ងន់លើមូលដ្ឋានគ្រឹះបច្ចេកទេសដែលចាំបាច់សម្រាប់ដំណើរការហាងអនឡាញ ហើយឯកសារដែលយកមកប្រើប្រាស់ក្នុងការធ្វើសារណានេះគឺចាប់ពីឆ្នាំ ២០១៣ ដល់ ២០២៤។

១.៥. សារៈសំខាន់នៃការសិក្សាស្រាវជ្រាវ

ការស្រាវជ្រាវនេះគឺមានសារៈសំខាន់ព្រោះ វាដោះស្រាយបញ្ហាប្រឈមដែលកំពុងប្រឈមមុខដោយ ផ្តល់បទពិសោធន៍ថ្មីដល់អ្នកប្រើប្រាស់ ដោយប្រើប្រាស់បច្ចេកវិទ្យាគេហទំព័រទំនើបជំនួយវិញ ។ ដោយការរួម បញ្ចូល React.js, Node.js, Express.js និង MongoDB ការស្រាវជ្រាវផ្តល់នូវដំណោះស្រាយដែលត្រូវបាន រចនាឡើងដើម្បីបង្កើនមុខងារ និងប្រសិទ្ធភាពនៃការទិញទំនិញអនឡាញ។ លើសពីនេះ ការស្រាវជ្រាវនេះគាំទ្រ សហគ្រាសធុនតូច និងមធ្យម (SMEs) ប្រកបដោយប្រសិទ្ធភាព ដែលអាចឱ្យពួកគេប្រកួតប្រជែងកាន់តែមាន ប្រសិទ្ធភាព ជាមួយសាជីវកម្មធំៗ។ ការសិក្សានេះក៏លើកកម្ពស់ការទទួលយកបច្ចេកវិទ្យាទំនើបៗ ដោយបង្ហាញ ពីអត្ថប្រយោជន៍ជាក់ស្តែង និងលើកទឹកចិត្តឱ្យមានការបង្កើតថ្មីបន្ថែមទៀតក្នុងការអភិវឌ្ឍន៍គេហទំព័រ។

សរុបមក សារៈសំខាន់នៃការស្រាវជ្រាវនេះគឺ គាំទ្រដល់កំណើនអាជីវកម្ម និងជំរុញការច្នៃប្រឌិតបច្ចេកវិទ្យា។

១.៦. វិធីសាស្ត្រនៃការសិក្សាស្រាវជ្រាវ

ដើម្បីសម្រេចបាននូវគោលបំណង វិធីសាស្ត្រខាងក្រោមនឹងត្រូវបានអនុម័ត៖

១.៦.១.ការរចនាប្រព័ន្ធ

សារណានេះ នឹងចាប់ផ្តើមជាមួយនឹងការចនាប្រព័ន្ធ ដោយកំណត់ពីសមាសធាតុផ្សេងគ្នា ផ្នែកខាងមុខ(Front-End) ផ្នែកខាងក្រោយ(Back-End) និងមូលដ្ឋានទិន្នន័យ(Data-Base)។

១.៦.២ ការអនុវត្ត

ដំណើរការអភិវឌ្ឍន៍នឹងត្រូវបានបែងចែកជាដំណាក់កាល រួមទាំងការអភិវឌ្ឍន៍ផ្នែកខាង មុខដោយប្រើ React.js ការអភិវឌ្ឍន៍ផ្នែកខាងក្រោយដោយប្រើ Node.js និង Express.js និង ការគ្រប់គ្រងមូលដ្ឋានទិន្នន័យ ដោយប្រើ MongoDB និង Mongoose ។

១.៦.៣. ការធ្វើតេស្ត និងការវាយតម្លៃ

ចុងក្រោយប្រព័ន្ធរបស់យើង នឹងត្រូវបានធ្វើតេស្តយ៉ាងម៉ត់ចត់ ដើម្បីធានាថាវាបំពេញតាម តម្រូវការ មុខងារ និងដំណើរការបានល្អក្រោមលក្ខខណ្ឌផ្សេងៗ។ មតិអ្នកប្រើក៏ នឹងត្រូវបាន ប្រមូលផងដែរ ដើម្បីវាយតម្លៃ លទ្ធភាពប្រើប្រាស់ និងបទពិសោធន៍ទូទៅ។

១.៧. រចនាសម្ព័ន្ធនៃការស្រាវជ្រាវ

នៅក្នុងការសរសេរសារណានេះត្រូវបានបែងចែកចេញជា ៥ ជំពូក ៖

ជំពូកទី១៖ ស្ថាបត្យកម្មប្រព័ន្ធ និងលំនាំរចនា (System Architecture and Design Patterns)

ជំពូកនេះស្វែងយល់អំពីការការអភិវឌ្ឍន៍ហាងអនឡាញដែលលើកយកអំពីប្រព័ន្ធទាំងមូលរួមទាំងធាតុ ផ្សំ ហើយក៏គ្របដណ្តប់លើគំរូនៃការចនាដែលប្រើដើម្បីដោះស្រាយបញ្ហាជាក់លាក់ ជំពូកនេះផ្តល់នូវការយល់ ដឹងអំពីការសម្រេចចិត្តលើបច្ចេកទេស និងវិធីសាស្ត្រដែលកំណត់រចនាសម្ព័ន្ធនៃកម្មវិធី។

ជំពូកទី២៖ ការរចនាប្រព័ន្ធ (System Design)

ជំពូកនេះរៀបរាប់លម្អិតអំពីការរចនានៃប្រព័ន្ធរបស់ហាងអនឡាញ រួមមានទាំងផ្នែកខាងមុខ(Front-End) និងផ្នែកខាងក្រោយ(Back-End)។ ផ្នែកនេះផ្តល់នូវប្លង់មេនៃរបៀបដែលប្រព័ន្ធត្រូវបានសាងសង់ និង របៀបដែលធាតុផ្សេងគ្នាធ្វើការជាមួយគ្នាដើម្បីសម្រេចបាននូវគោលបំណងរបស់គម្រោង។

ជំពូកទី៣៖ ការពិចារណាអំពីសុវត្ថិភាព (Security Considerations)

ជំពូកនេះនឹងផ្តោតលើវិធានការសុវត្ថិភាពដែលបានអនុវត្តនៅក្នុងហាងអនឡាញរបស់អ្នក។ វានឹងដោះ ស្រាយទិដ្ឋភាពផ្សេងៗនៃសុវត្ថិភាព ដូចជាការពារទិន្នន័យអ្នកប្រើ ធានាសុវត្ថិភាព API endpoint និងធានា

ប្រតិបត្តិការសុវត្ថិភាព។ វាក៏នឹងគ្របដណ្តប់លើការអនុវត្តល្អបំផុត និងបច្ចេកវិទ្យាដែលប្រើដើម្បីកាត់បន្ថយហានិភ័យសុវត្ថិភាពផងដែរ។

ជំពូកទី៤៖ បទពិសោធន៍អ្នកប្រើប្រាស់ និងការរចនា (User Experience and Design)

ជំពូកនេះផ្តោតលើការវាយតម្លៃបទពិសោធន៍អ្នកប្រើប្រាស់នៃហាងអនឡាញ។ ជាវិធីសាស្ត្រសាកល្បងអ្នកប្រើប្រាស់ និង ការប្រមូលមតិកែលម្អ ។ ជំពូកនេះ ពិនិត្យមើលពីរបៀបដែលអ្នកប្រើប្រាស់ ធ្វើអន្តរកម្ម (Interact)ជាមួយប្រព័ន្ធនិងរបៀបដែលមតិកែលម្អរបស់ពួកគេត្រូវបានប្រើប្រាស់ដើម្បីធ្វើឱ្យប្រសើរឡើង។វាមានគោលបំណងធានាថាហាងអនឡាញបំពេញតម្រូវការអ្នកប្រើប្រាស់និងផ្តល់នូវបទពិសោធន៍ល្អដល់អ្នកប្រើ ។

ជំពូកទី៥៖ លទ្ធផលនៃការស្រាវជ្រាវ

ជំពូកនេះសង្ខេបលទ្ធផលស្រាវជ្រាវ និងឆ្លុះបញ្ចាំងពីភាពជោគជ័យនៃគម្រោង ថាតើគម្រងដែលយើងបានធ្វើទទួលបានអ្វីខ្លះ សម្រេចបានប៉ុន្មានភាគរយ និងបង្ហាញពីដំណើរការផ្សេងៗដែលប្រព័ន្ធរបស់យើងបានធ្វើកន្លងមកជួយបានអ្វីខ្លះដល់អ្នកប្រើប្រាស់ប្រព័ន្ធរបស់យើង។

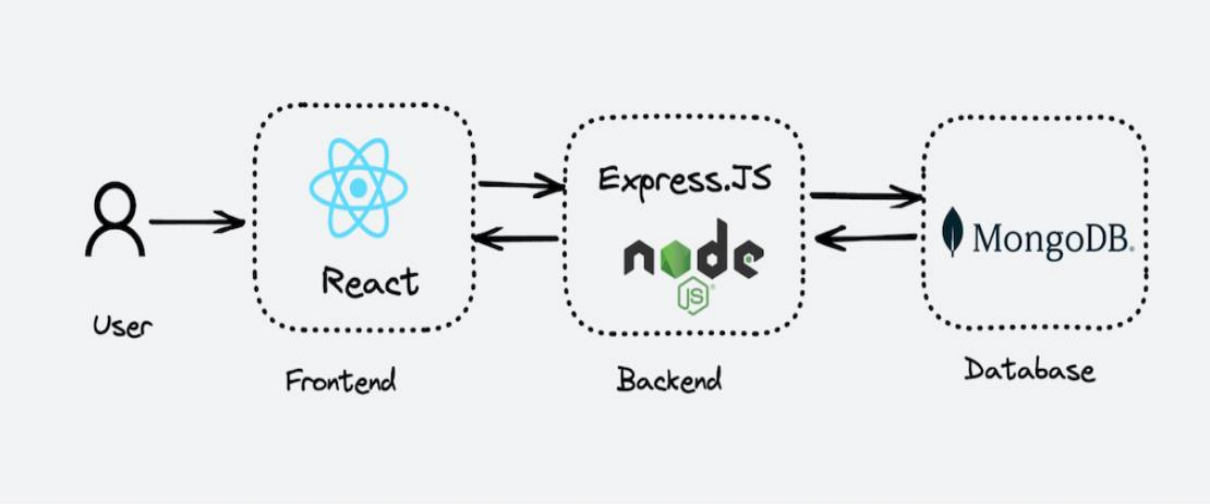
ជំពូកទី ១

ស្ថាបត្យកម្មប្រព័ន្ធ និងលំនាំរចនា

(System Architecture and Design Patterns)

១.១. និដ្ឋភាពទូទៅនៃស្ថាបត្យកម្មប្រព័ន្ធ (Overview of System Architecture)

ស្ថាបត្យកម្មប្រព័ន្ធ (System Architecture) គឺជាប្លង់មេដែលកំណត់ការរចនាសម្ព័ន្ធ និងតរិយោបថនៃកម្មវិធីហាងអនឡាញ។ កម្មវិធីប្រើ **MERN (MongoDB, Express.js, React.js, Node.js)** ជាស្ថាបត្យកម្មស្នូល។ ជម្រើសនេះអនុញ្ញាតឱ្យមានការរួមបញ្ចូលគ្នានៃបច្ចេកវិទ្យាទាំងផ្នែកខាងមុខ **(Front-End)**និងផ្នែកខាងក្រោយ**(Back-End)** ដោយប្រើប្រាស់ **JavaScript** ដែលជួយសម្រួលដល់ដំណើរការអភិវឌ្ឍន៍។



រូបភាពទី១.១៖ MERN stack architecture

១.១.១. MongoDB

MongoDB គឺជាមូលដ្ឋានទិន្នន័យ NoSQL ត្រូវបានប្រើដើម្បីរក្សាទុកទិន្នន័យព័ត៌មានអ្នកប្រើ ប្រាស់ និងព័ត៌មានលម្អិតនៃការបញ្ជាទិញរបស់អ្នកប្រើប្រាស់ ។

១.១.២. Express.js

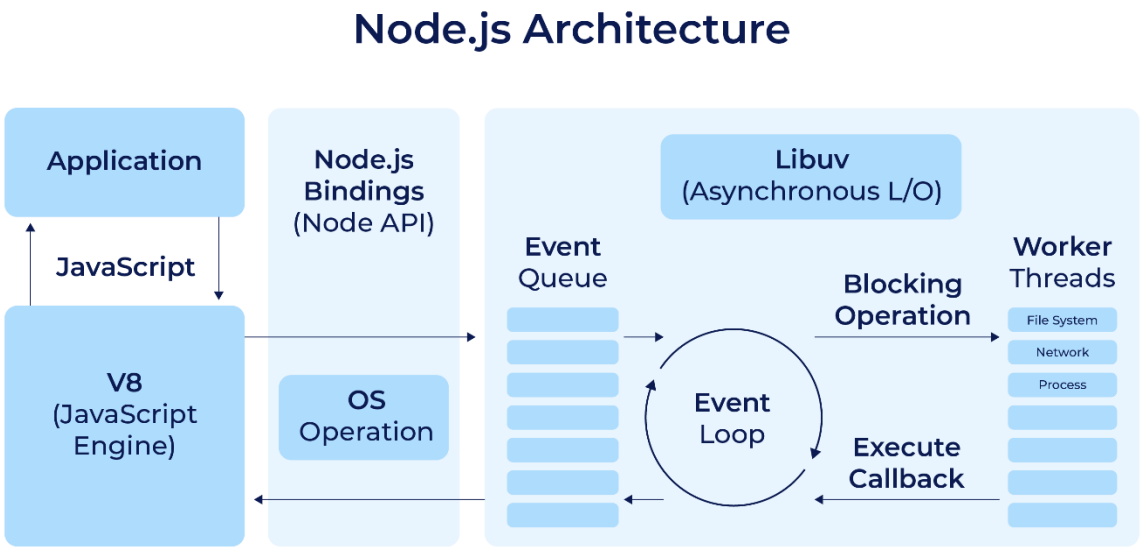
Express.js គឺជា framework របស់ Node.js ដែលប្រើលើការភ្ជាប់ផ្នែកខាងមុខ **(Front-End)** ទៅផ្នែកខាងក្រោយ **(Back-End)** ដែលមានន័យថា **API (Application Programming Interface)**។

១.១.៣. React.js

React.js គឺជា Library របស់ JavaScript សម្រាប់ការបង្កើតផ្នែកខាងមុខ **(Front-End)** នៃកម្មវិធី។

១.១.៤. Node.js

Node.js គឺជាភាសាដែលយកមកប្រើផ្នែកខាងក្រោយ (Back-End) របស់ JavaScript ។



រូបភាពទី ១.១.៤៖ បង្ហាញពី Nodejs Work

១.២. ស្ថាបត្យកម្មផ្នែកខាងមុខ (Frontend Architecture React.js)

ផ្នែកខាងមុខនៃ ហាងអនឡាញ **(Online Shop)** ត្រូវបានបង្កើតឡើងដោយប្រើប្រាស់ **React.js** ដែលជា **Library JavaScript** ដ៏មានអានុភាពដែលត្រូវបានរចនាឡើងសម្រាប់បង្កើត interactive និង reusable សម្រាប់អ្នកប្រើ ។ ផ្នែកនេះគ្របដណ្តប់លើស្ថាបត្យកម្មជាមូលដ្ឋាននៃផ្នែកខាងមុខរបស់ React.js ។

១.២.១. React Component-Based Architecture

React គឺផ្តោតទៅលើសមាសធាតុ ដែលជាបណ្តុំនៃ UI (Block of UI) ។ សមាសធាតុនីមួយៗតំណាងឱ្យផ្នែកជាក់លាក់នៃកម្មវិធី ដូចជាការកន្លែងទិញផលិតផល ឬទំព័របង់ប្រាក់ចេញ(Checkout)។ ទោះជាយ៉ាងណាក៏ដោយ ចាប់តាំងពីការដាក់បញ្ចូល React hooks ឥឡូវនេះ សមាសធាតុមុខងារអាច

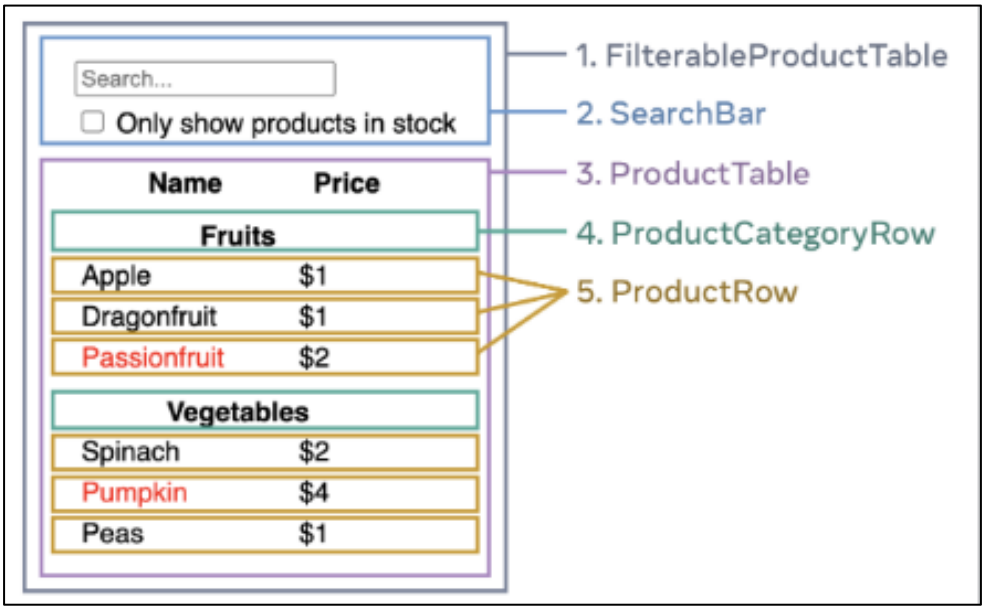
គ្រប់គ្រងស្ថានភាពក្នុងគេហទំព័របាន ដែលធ្វើឱ្យពួកវាកាន់តែមានប្រជាប្រិយភាព(Popular)នៅក្នុងការអភិវឌ្ឍន៍ React ។

១.២.១.១. សមាសធាតុដែលអាចប្រើឡើងវិញបាន (Component Reusability)

React លើកកម្ពស់លទ្ធភាពប្រើប្រាស់ឡើងវិញ ដោយអនុញ្ញាតឱ្យសមាសធាតុ ទៅជាបណ្តុំតូចៗ (Block of Code) និងឯករាជ្យ (Independently) ។ សមាសធាតុដូចជា ProductItem (ដែលបង្ហាញលិទ្ធផលដូចៗគ្នា) អាចត្រូវបានប្រើប្រាស់ឡើងវិញនៅក្នុងផ្នែកផ្សេងៗនៃកម្មវិធី ដូចជាទំព័រលទ្ធផលស្វែងរក ឬផ្នែកណែនាំផលិតផល។

១.២.១.២. ឋានានុក្រមសមាសភាគ (Component Hierarchy)

កម្មវិធីត្រូវបានបែងចែកជាផ្នែកតូចៗដែលបង្កើតដើម្បីគ្រប់គ្រង UI ឱ្យមានប្រសិទ្ធភាព។ សមាសភាគនីមួយៗទទួលខុសត្រូវក្នុងការបង្ហាញផ្នែកជាក់លាក់នៃ UI (User Interface) ។



រូបភាពទី ១.២.១.២៖ បង្ហាញពី Component Hierarchy

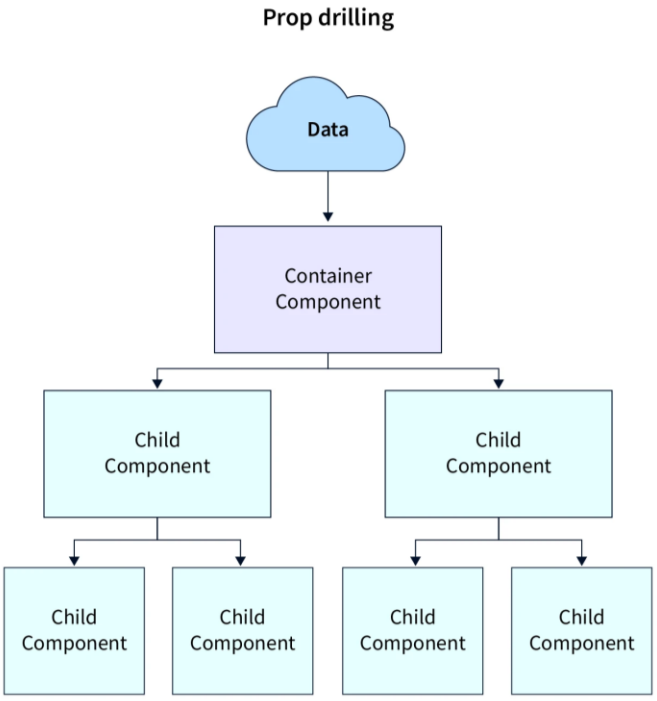
១.៣. ការគ្រប់គ្រងទិន្នន័យក្នុងប្រព័ន្ធ (State Management in React)

State តំណាងឱ្យទិន្នន័យដែលកំពុងប្រើប្រាស់នៅក្នុងប្រព័ន្ធ និងមានឥទ្ធិពលលើរបៀបដែលសមាសធាតុប្រព្រឹត្ត និងបង្ហាញ។ ឧទាហរណ៍ ថាទិន្នន័យ ចំនួនផលិតផល និងតម្លៃសរុបដែលបានដាក់ចូលទៅក្នុង Cart ក្នុងករណីខ្លះក្នុងប្រព័ន្ធរបស់យើងត្រូវការចែករំលែកទិន្នន័យដូចគ្នា ដូចនេះយើងអាចប្រើ State។

ឧទាហរណ៍៖ State: const [cartItems, setCartItems] = useState([]);

១.៣.១. Context API in React

Context API ក៏ជាប្រភេទ State មួយផងដែរតែគេចាត់ទុកវាជា global state ដែលអាចហៅទៅប្រើបាននៅក្នុង project យើងទាំងមូលបានដោយមិនចាប់បាច់ការបោះ props ពី component មួយទៅ component មួយទៀតទេ គឺយើងអាចហៅ Context ក្នុង component របស់យើងតែម្តង។



រូបភាពទី ១.៣.១៖ បង្ហាញពី Prop drilling

១.៤. React Routing (React Router)

សម្រាប់កម្មវិធីមួយទំព័រ (Single Page Applications) React Router អាចអោយយើងទៅទំព័រផ្សេងៗគ្នា ដោយមិនចាំបាច់លោតទំព័រទាំងមូលឡើងវិញ។ នៅក្នុងហាងអនឡាញ ការទៅទំព័រផ្សេងមួយទៀតមានសារៈសំខាន់ណាស់ នៅពេលដែលអ្នកប្រើប្រាស់ផ្លាស់ទីរវាងបញ្ជីផលិតផល(product listings)ទំព័រផលិតផលនីមួយៗ(individual product pages) និងទំព័របង់ប្រាក់ចេញ(checkout page)។

១.៤.១. BrowserRouter

BrowserRouter ប្រើដើម្បីក្តោប**(wraps)** កម្មវិធីទាំងមូល ហើយBrowserRouter មានតួនាទីសង្កេត

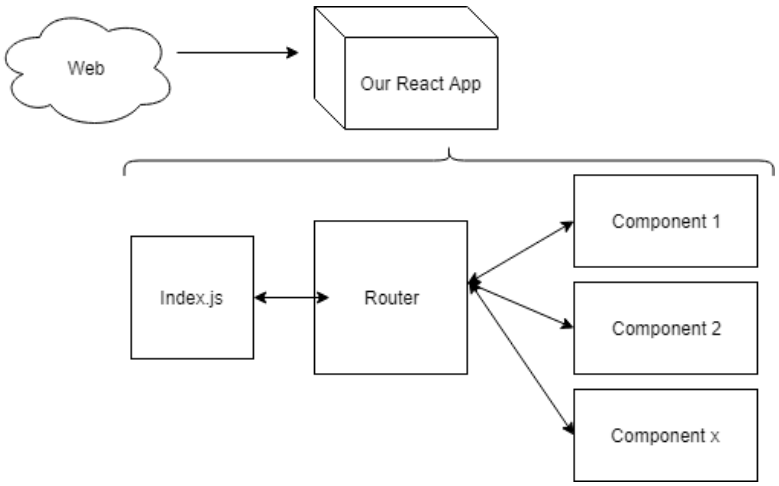
ការផ្លាស់ប្តូរ URL (Uniform Resource Locator) ដើម្បីប្តូរទំព័រ (Web Page) ។

១.៤.១.១. Route

Route ប្រើដើម្បីប្តូរទំព័រ (Page) ទៅតាម URL ដែលទទួលបានពី BrowserRouter នៅពេលដែរអ្នកប្រើបានប្តូរទៅទំព័រណាមួយថ្មី ។

១.៤.១.២. Link

Link ត្រូវបានប្រើជំនួសអោយ (<a>) នៅក្នុងHTML តែក្នុងReact ប្រើ <Link> សម្រាប់ការប្តូរទំព័រណាមួយ។ ទាំងនេះអនុញ្ញាតឱ្យកម្មវិធីផ្លាស់ប្តូរការមើលទំព័រ Web Page ដោយមិនចាំបាច់លោតទំព័រទាំងអស់ឡើងវិញ ដែលធ្វើអោយបទពិសោធន៍ (UX) អ្នកប្រើប្រាស់ប្រសើរឡើង។



រូបភាពទី ១.៤.២៖ បង្ហាញពី React Router

១.៥. User Interface និងការរចនា (User Interface and Styling with MUI)

នៅក្នុងការអភិវឌ្ឍន៍ហាងអនឡាញនេះ Material UI (MUI) ត្រូវបានប្រើជាក្របខ័ណ្ឌ UI ចម្បង។

MUI គឺជាLibrary ដ៏ពេញនិយមដែលផ្តល់នូវសំណុំពេញលេញនៃសមាសធាតុ React ដែលបង្កើតជាមុនដែលអាចអោយយើងធ្វើការប្តូរបាន ។ នេះជួយសម្រួលដំណើរការរចនា ខណៈពេលដែលធានានូវការរចនាស្របគ្នា នៅកម្មវិធីទាំងមូល។

១.៥.១. សមាសធាតុ MUI (MUI Components)

MUI ផ្តល់សមាសធាតុជាច្រើន ដូចជា buttons, forms, cards, navigation bars, and grids ដែលអាចអោយយើងប្រើជាមួយនឹងការរចនាផ្នែកខាងមុខ (Front-End) របស់យើងបានលឿនផងដែរ។

```
import { Button, Card, CardContent, Typography } from '@mui/material';

const ProductCard = ({ product }) => (
  <Card>
    <CardContent>
      <Typography variant="h5">{product.name}</Typography>
      <Typography variant="body2" color="textSecondary">
        {product.description}
      </Typography>
      <Typography variant="h6">${product.price}</Typography>
      <Button variant="contained" color="primary">
        Add to Cart
      </Button>
    </CardContent>
  </Card>
);
```

១.៥.២. រូបរាង និងការកែប្រែជាមួយ MUI (Theming and Customization with MUI)

MUI Themes ផ្តល់នូវវិធីងាយស្រួលក្នុងការកែប្រែរូបរាង ទាំងមូលក្នុងកម្មវិធី។ អាចកំណត់ពណ៌ផ្ទាល់ខ្លួន ពុម្ពអក្សរ ទំហំអក្សរ និងគម្លាតផ្សេងៗតាមការចង់បានរបស់យើង។

```
import { createTheme, ThemeProvider } from '@mui/material/styles';

const theme = createTheme({
```

```
  palette: {
    primary: {
      main: '#ff5733', // custom primary color
    },
    secondary: {
      main: '#333333', // custom secondary color
    },
  },
  typography: {
    fontFamily: 'Roboto, Arial, sans-serif',
    h5: {
      fontWeight: 600,
    },
  },
});
const App = () => (
  <ThemeProvider theme={theme}>
    {/* Application components using the custom theme */}
  </ThemeProvider>);
```

ជាមួយនឹងការរៀបចំរចនាបទនេះ សមាសធាតុ MUI ទាំងអស់នៅក្នុងកម្មវិធីនឹងប្រើពណ៌ ពុម្ពអក្សរ និង

រចនាបទដែលបានកំណត់ ដូចគ្នាទាំងអស់នៅក្នុងកម្មវិធីទាំងមូល ។

១.៥.៣. Grid Layout for Responsive Design

Grid Layout របស់ MUI អនុញ្ញាតឱ្យអ្នកអភិវឌ្ឍន៍បង្កើតWeb Responsive និងធ្វើអោយត្រូវគ្រប់ទំហំអេក្រង់។ នេះជានាថាហាងអនឡាញគឺងាយស្រួលប្រើសម្រាប់ទូរស័ព្ទ និងផ្តល់នូវបទពិសោធន៍អ្នកប្រើប្រាស់ដ៏ល្អនៅលើឧបករណ៍ផ្សេងៗគ្នា។

ឧទាហរណ៍៖ Grid Layout for Product Listing

```
import { Grid } from '@mui/material';

const ProductList = ({ products }) => (
 <Grid container spacing={2}>
  {products.map((product) => (
   <Grid item xs={12} sm={6} md={4} lg={3} key={product.id}>
    <ProductCard product={product} />
   </Grid>
  ))}
 </Grid>);
```

រូបភាពទី ១.៥.៣៖ បង្ហាញពី Grid Layout for Responsive

ជំពូកទី ២

ការរចនាប្រព័ន្ធ (System Design)

ជំពូកនេះបង្ហាញអំពីការរចនានៃប្រព័ន្ធទិញទំនិញអនឡាញដែលមានគោលបំណងអនុញ្ញាតឱ្យអ្នកប្រើប្រាស់រុករក បញ្ជាទិញ និងគ្រប់គ្រងផលិតផលប្រកបដោយប្រសិទ្ធភាព។ ប្រព័ន្ធនេះរួមបញ្ចូលផ្នែកខាងមុខ React.js ផ្នែកខាងក្រោយ Node.js ជាមួយ Express.js និងមូលដ្ឋានទិន្នន័យ MongoDB ។ ការរចនាផ្តោតលើលទ្ធភាពធ្វើមាត្រដ្ឋាន(Scalability) សុវត្ថិភាព និងភាពងាយស្រួលនៃការប្រើប្រាស់ ដោយដោះស្រាយតម្រូវការផ្សេងៗ ដើម្បីធានាបាននូវបទពិសោធន៍ទិញទំនិញអនឡាញដ៏រឹងមាំ និងងាយស្រួលប្រើ។

២.១. ទិដ្ឋភាពទូទៅនៃប្រព័ន្ធ

ហាងអនឡាញត្រូវបានបង្កើតឡើងដោយប្រើប្រាស់ Client-Server Model ដែលផ្នែកខាងមុខ(Front-End) ទាក់ទងជាមួយផ្នែកខាងក្រោយ(Back-End)តាមរយៈ API Request ហើយផ្នែកខាងក្រោយធ្វើអន្តរកម្ម (Interactive) ជាមួយមូលដ្ឋានទិន្នន័យ MongoDB ដើម្បីទាញយក ឬរក្សាទុកទិន្នន័យ។

២.១.១. សមាសធាតុ

- **Frontend (React.js with MUI) :** ប្រើដើម្បី ផ្ញើសំណើ API ទៅផ្នែកខាងក្រោយ(Back-End)។

- **Backend (Node.js, Express.js):** ប្រើដើម្បីទទួលការផ្ញើសំណើ API ពី Front-end ធ្វើការទាក់ទងជាមួយ database ហើយបញ្ជូនទន្ទិន័យពី Database ទៅអោយអ្នកប្រើប្រាស់វិញតាមអ្វីដែលអ្នកប្រើចង់បាន។

- **Database (MongoDB):** ជាកន្លែងដែលផ្ទុកទិន្នន័យដែលពាក់ពន្ធនឹងអ្នកប្រើប្រាស់ ផលិតផល, ប្រភេទផលិតផល, ការបញ្ជាទិញ និង ផ្សេងៗទៀត។

២.១.២. លំហូរទិន្នន័យ (Data Flow)

- អ្នកប្រើប្រាស់ធ្វើទំនាក់ទំនងនៅលើផ្នែកខាងមុខ(Front-End) (ឧទាហរណ៍. ការមើលផលិតផល ការបញ្ចូលទៅក្នុងCart ការដាក់ការបញ្ជាទិញ)។

- ផ្នែកខាងមុខផ្ញើសំណើ HTTP ទៅផ្នែកខាងក្រោយ (ឧទាហរណ៍. ដើម្បី ទាញយកទិន្នន័យ

ផល ឬបង្កើតការបញ្ជាទិញ)។

- កម្មវិធីខាងក្រោយ (Back-End)ដំណើរការសំណើរ សាកសួរមូលដ្ឋានទិន្នន័យ (queries) MongoDB និងផ្ញើការឆ្លើយតបមកវិញ។

- ផ្នែកខាងមុខធ្វើបច្ចុប្បន្នភាព (Update) UI ដោយផ្អែកលើការឆ្លើយតបដែលទទួលបានពីផ្នែកខាងក្រោយ (Back-End) ។

២.២. រចនាបទផ្នែកខាងមុខ (Frontend Design)

ផ្នែកខាងមុខត្រូវបានបង្កើតឡើងដោយប្រើប្រាស់ React.js និងរចនាជាមួយ Material UI (MUI) សម្រាប់ការរចនាទាន់សម័យ អាចប្រើប្រាស់បានគ្រប់ទំហំ។

២.២.១. សមាសធាតុសំខាន់ៗផ្នែក Frontend

- **ទំព័រផលិតផល:** បង្ហាញផលិតផលទាំងអស់នៅលើទំព័រ ដែលផលិតផលនីមួយៗមានភ្ជាប់ជាមួយឈ្មោះ តម្លៃ រូបភាព និង ប៊ូតុងចុចដើម្បីដាក់ product ចូលក្នុង Cart ។

- **Shopping Cart:** អនុញ្ញាតឱ្យអ្នកប្រើប្រាស់មើលទំនិញដែលពួកគេបានបញ្ចូលទៅក្នុង Cart របស់ពួកគេ កែតម្រូវបរិមាណ ឬដកចេញ។

- **User Authentication:** អ្នកប្រើប្រាស់អាចចុះឈ្មោះ ចូល និងចេញ។ Routeត្រូវបានការពារ (Protected routes) ដើម្បីធានាបានថាមានតែអ្នកប្រើប្រាស់ដែលបានចូលប៉ុណ្ណោះដែលអាចធ្វើការបញ្ជាទិញបាន។

២.៣. រចនាបទផ្នែកខាងក្រោយ (Backend Design)

ផ្នែកខាងក្រោយត្រូវបានដំណើរការដោយ Node.js ជាមួយនឹងFramework Express.js ដែលផ្តល់នូវដំណោះស្រាយផ្នែកខាង server-side ដែលមានល្បឿនលឿន និងអាចធ្វើមាត្រដ្ឋាន (scalable) បាន។ It handles the business logic និងធ្វើការប្រាស្រ័យទាក់ទងជាមួយទិន្នន័យដែលផ្ទុកនៅក្នុង Database។

២.៣.១. សមាសធាតុសំខាន់ៗផ្នែក Backend

- **API Routing:** ផ្នែកខាងក្រោយ (backend) ប្រើ RESTful API សម្រាប់ប្រតិបត្តិការផ្សេងៗគ្នា៖

- POST api/login : គឺជាApi requestដើម្បីអោយអ្នកប្រើប្រាស់អាចចូលទៅប្រើប្រាស់។

- POST api/signup: គឺជាApi requestដើម្បីអាចអោយអ្នកប្រើប្រាស់អាចបង្កើត ឬចុះឈ្មោះក្នុងការប្រើប្រាស់កម្មវិធី។

- POST /api/order : គឺជាApi requestដើម្បីអោយអ្នកប្រើប្រាស់អាចធ្វើការកម្ម៉ង់ទំនិញ។

- GET /api/product : គឺជាApi requestដើម្បីផ្តល់ចំនួនផលិតផលទាំងអស់ដែលមានទៅអោយអ្នកប្រើប្រាស់បានឃើញ។

- POST /api/product: គឺជាApi requestដើម្បីអាចអោយអ្នកលក់អាចបង្កើតផលិតផលថ្មី។

- GET /api/category: គឺជាApi requestដើម្បីផ្តល់ប្រភេទផលិតផលទាំងអស់ដែរមានទៅអោយអ្នកប្រើប្រាស់។

- POST /api/category: គឺជាApi requestដើម្បីអោយអ្នកលក់ ឬក៏ម្ចាស់អាជីវកម្មអាចធ្វើការបង្កើតប្រភេទទំនិញថ្មី។

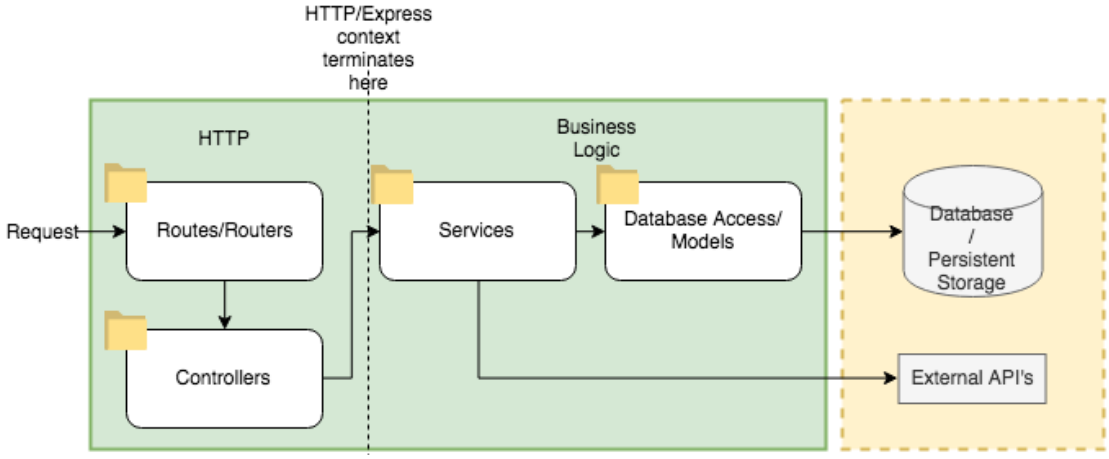
- POST api/promotion: គឺជាApi requestដើម្បីអោយអ្នកលក់អាចបង្កើតការបញ្ចុះតម្លៃទៅលើផលិតផលណាមួយ។

- GET api/promotion: គឺជាApi requestដើម្បីផ្តល់ទំនិញដែលមានការបញ្ចុះតម្លៃទៅអោយអ្នកប្រើប្រាស់បានឃើញ។

- **គ្រឿងកណ្តាល (Middleware):** ប្រើដើម្បីការកំណត់សម្គាល់ទៅលើអ្វីមួយដូចជាការចូលប្រើប្រាស់កម្មវិធីរបស់អ្នកប្រើមានភាពត្រឹមត្រូវដែរ ឬទេពេលដែលអ្នកប្រើប្រាស់បានចូលប្រើ មិនតែប៉ុណ្ណោះ Middleware ក៏ប្រើសម្រាប់គ្រប់គ្រងការមានបញ្ហាផ្សេងៗដូចជា api error, data validation ។

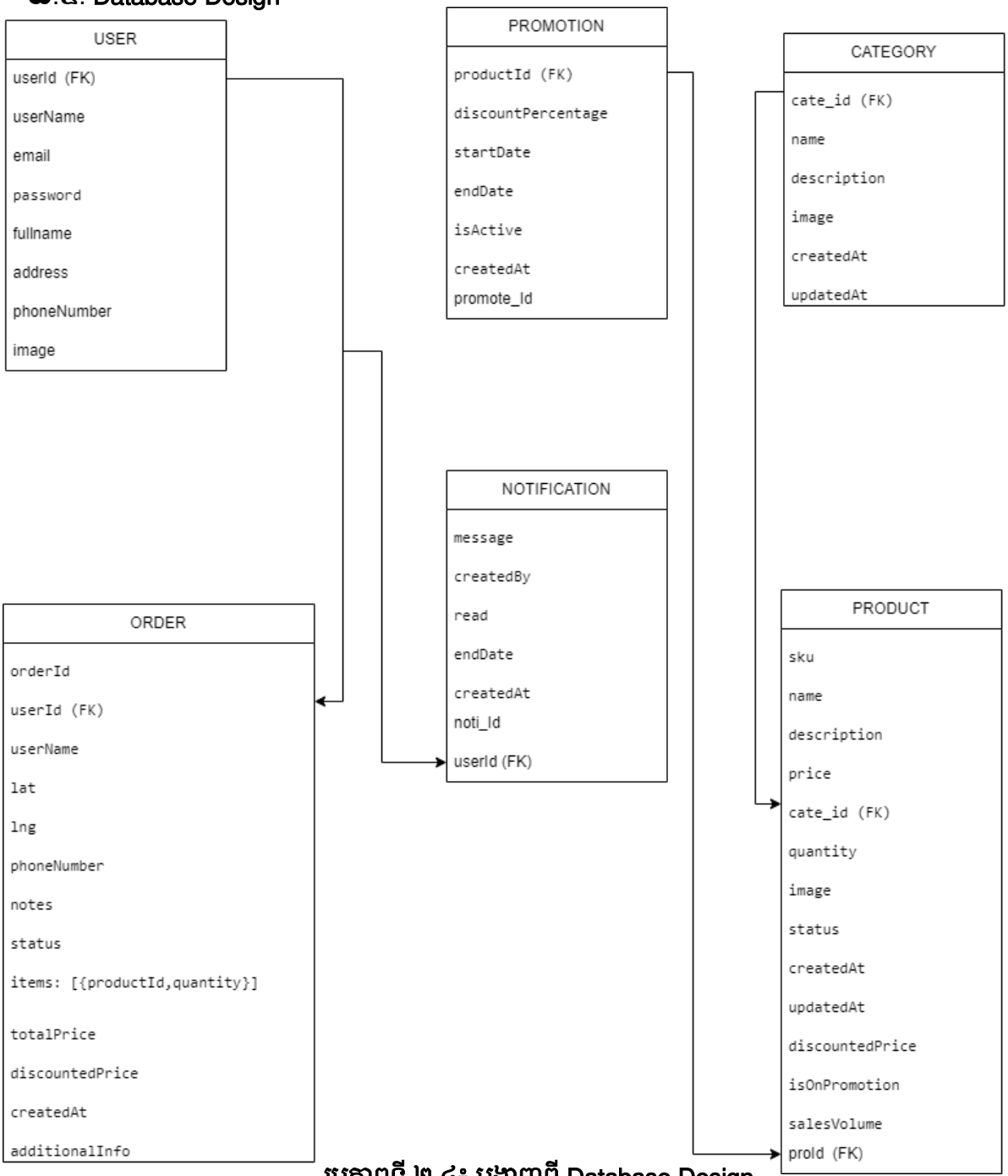
- ការធ្វើទំនាក់ទំនងជាមួយកន្លែងផ្ទុកទិន្នន័យ (Database Interaction): The backend ប្រើ

Mongoose, an Object Data Modeling (ODM) library, ដើម្បីធ្វើការទំនាក់ទំនងជាមួយ MongoDB. គំរូ Mongoose ប្រើដើម្បីកំណត់ទម្រង់ទិន្នន័យផ្សេងៗដែលមានទម្រង់ខុសៗពីគ្នាដើម្បីធ្វើការផ្ទុកទិន្នន័យទាំងនោះនៅក្នុង Database។



រូបភាពទី ២.១.១៖ បង្ហាញពី សមាសធាតុសំខាន់ៗផ្នែក Backend

២.៤. Database Design



រូបភាពទី ២.៤៖ បង្ហាញពី Database Design

មូលដ្ឋានទិន្នន័យ MongoDB រក្សាទុកទិន្នន័យសំខាន់សម្រាប់ហាងអនឡាញ រួមទាំងអ្នកប្រើប្រាស់ ផលិតផល និងការបញ្ជាទិញ។ Collectionនីមួយៗធ្វើតាមគ្រោងការណ៍ជាក់លាក់ដែលកំណត់ដោយ Mongoose ។

២.៤.១. Collections សំខាន់ៗ

Collections សំដៅទៅលើបណ្តុំទិន្នន័យដែលមាន Field ផ្សេងៗគ្នា ដែលដូចគ្នាទៅនឹង Table នៅក្នុង Relational Database ។

២.៤.១.១. User Collection

នៅក្នុង User Collection នេះមានផ្ទុកទិន្នន័យដូចជា៖ username,email,password fullname, address, phoneNumber, proImage ។

២.៤.១.២. Product Collection

នៅក្នុង Product Collection នេះមានផ្ទុកទិន្នន័យដូចជា៖ sku,name,description,price,cate_id,quantity,image,status,createdAt,updatedAt discountedPrice,isOnPromotion,salesVolume ។

២.៤.១.៣. Category Collection

នៅក្នុង Category Collection នេះមានផ្ទុកទិន្នន័យដូចជា៖ cate_id, name, description, cateImage, createdAt, updatedAt។

២.៤.១.៤. Promotion Collection

នៅក្នុង Promotion Collection នេះមានផ្ទុកទិន្នន័យដូចជា៖ productId, discountPercenttage, startDate, endDate, isActive, createAt, updateAt ។

២.៤.១.៥. Order Collection

នៅក្នុង Promotion Collection នេះមានផ្ទុកទិន្នន័យដូចជា៖ userId, userName lat, lng, phoneNumber, notes, status, items, totoalPrice, additional ។

២.៤.១.៦. Notification Collection

នៅក្នុង Promotion Collection នេះមានផ្ទុកទិន្នន័យដូចជា៖ Message, createdBy, read, createdAt ។

ជំពូកទី៣

ការពិចារណាអំពីសុវត្ថិភាព (Security Considerations)

នៅក្នុងកម្មវិធីគេហទំព័រណាមួយ សុវត្ថិភាពគឺមានសារៈសំខាន់បំផុត ជាពិសេសសម្រាប់ហាងអនឡាញ ដែលទិន្នន័យអ្នកប្រើប្រាស់សំខាន់ៗ ដូចជាព័ត៌មានផ្ទាល់ខ្លួន និងព័ត៌មានលម្អិតអំពីការទូទាត់។ ជំពូកនេះរៀបរាប់អំពីវិធានការសុវត្ថិភាព ដែលត្រូវបានអនុវត្តនៅក្នុងប្រព័ន្ធ ដើម្បីការពារប្រឆាំងនឹងការគំរាមកំហែងដែលអាចកើតមាន ធានាភាពឯកជនទិន្នន័យ (Privacy Data) និងរក្សាបាននូវភាពត្រឹមត្រូវនៃប្រព័ន្ធ (maintain system integrity)។

៣.១. ការផ្ទៀងផ្ទាត់ និងការអនុញ្ញាត (Authentication and Authorization)

ការផ្ទៀងផ្ទាត់ភាពត្រឹមត្រូវ សំដៅទៅលើដំណើរការនៃការផ្ទៀងផ្ទាត់អត្តសញ្ញាណរបស់អ្នកប្រើប្រាស់។ ម៉្យាងទៀតការអនុញ្ញាត ធានាថាអ្នកប្រើប្រាស់មានការអនុញ្ញាតចាំបាច់ដើម្បីអនុវត្តសកម្មភាពជាក់លាក់នៅក្នុងប្រព័ន្ធ។

- **JWT Tokens (JSON Web Tokens) :** JWT ត្រូវបានប្រើក្នុងការបញ្ជាក់អត្តសញ្ញាណរបស់អ្នកប្រើប្រាស់នៅពេលដែលអ្នកប្រើប្រាស់ធ្វើការ Login ពេលដែល Login បានជោគជ័យ backend និង ផ្តល់ជា JWT Tokens ដើម្បីធ្វើការបញ្ជាក់ថាពិតជា អ្នកប្រើប្រាស់ពិតមែន។

- **Secure Routes :** Routes GET /api/orders (សម្រាប់ការមើលការបញ្ជាទិញ) ឬ POST /api/products (សម្រាប់ការបន្ថែមផលិតផលថ្មី សម្រាប់ Admin ប៉ុណ្ណោះ) route ទាំងនេះត្រូវការ Token ក្នុងការ perform ការងារដែលពាក់ព័ន្ធនឹង route ទាំងនេះ។ ប្រសិនបើមិនមាន Tokens ឬTokensមិនត្រឹមត្រូវ អ្នកប្រើប្រាស់មិនអាចធ្វើការណាដែលត្រូវការ Tokens នោះទេ។

- **Role-based Access Control:** តួនាទីផ្សេងៗគ្នា **(admin,user)** ត្រូវបានកំណត់នៃការចូលប្រើប្រព័ន្ធផ្សេងៗគ្នា។ អ្នកគ្រប់គ្រងអាចគ្រប់គ្រងផលិតផល ប្រភេទ និងការបញ្ជាទិញបាន ខណៈពេលដែលអ្នកប្រើប្រាស់ធម្មតាអាចធ្វើ Interactive ជាមួយគណនី និងការបញ្ជាទិញរបស់ពួកគេតែប៉ុណ្ណោះ។

៣.២. ការអ៊ិនគ្រីបទិន្នន័យ (Data Encryption)

ព័ត៌មានដែលសំខាន់ៗ ត្រូវតែត្រូវបានការពារ ដើម្បីការពារការចូលប្រើដោយគ្មានការអនុញ្ញាត។ ការអ៊ិនគ្រីបទិន្នន័យ (Data Encryption) ដើរតួនាទីយ៉ាងសំខាន់ក្នុងការការពារអត្តសញ្ញាណអ្នកប្រើប្រាស់ និងទិន្នន័យឯក

ឯកជនណាមួយដែលបានបញ្ជូនតាមអ៊ីនធឺណេត។

៣.២.១. ការអ៊ិនគ្រីបពាក្យសម្ងាត់ (Encryption of Passwords)

- **bcrypt៖** លេខសម្ងាត់របស់អ្នកប្រើត្រូវបាន hash ដោយប្រើ bcrypt ដែលជាក្បួនដោះស្រាយ hashing សុវត្ថិភាព។ វាធានាថា ទោះប្បីជាមានអ្នកព្យាយាមចូលគណនីរបស់យើង ក៏ពិបាកដែល ព្រោះថា ពេលដែលពាក្យសម្ងាត់ត្រូវបាន encrypt ពាក្យសម្ងាត់មិនអាចត្រូវបានរកឃើញវិញបានយ៉ាងងាយស្រួលនោះទេ។

- **Salting:** ជា random number បន្ថែមទៅពាក្យសម្ងាត់នីមួយៗ មុនពេល hash ដើម្បីពង្រឹងការអ៊ិនគ្រីបបន្ថែមទៀត។ នេះធានាថាទោះបីជាអ្នកប្រើប្រាស់ ពីរនាក់មានពាក្យសម្ងាត់ដូចគ្នាក៏ដោយ តម្លៃនៃការ hash របស់ពួកគេនឹងខុសគ្នាដែរ។

៣.២.២.ការបញ្ជូនទិន្នន័យប្រកបដោយសុវត្ថិភាព(Secure Transmission)

កម្មវិធីប្រើប្រាស់ **HTTPS (Hypertext Transfer Protocol Secure)** ដើម្បីអ៊ិនគ្រីបទិន្នន័យដែលបានបញ្ជូនរវាងផ្នែកខាងមុខ **(Front-End)** និងផ្នែកខាងក្រោយ **(Back-End)** ដោយការពារព័ត៌មានសំខាន់ៗ ដូចជាព័ត៌មានលម្អិតអំពីការបង់ប្រាក់អំឡុងពេលធ្វើការបង់ប្រាក់តាម Online។

៣.៣. Securing APIs

APIs គឺអនុញ្ញាតឱ្យផ្នែកខាងមុខ **(Front-End)** ទំនាក់ទំនងជាមួយផ្នែកខាងក្រោយ **(Back-End)** ។ **Securing APIs** គឺមានសារៈសំខាន់ដើម្បីការពារការចូលប្រើដោយគ្មានការអនុញ្ញាត និងការបាត់បង់ទិន្នន័យ។

៣.៣.១. API Authentication

- **Bearer Tokens:** សំណើ API ទាំងអស់ត្រូវបានផ្ទៀងផ្ទាត់ដោយប្រើ JWT bearer tokens។ និមិត្តសញ្ញាត្រូវតែបញ្ចូលទៅក្នុងសំណើ ដើម្បីធានាថា មានតែអ្នកប្រើប្រាស់ដែលមានការអនុញ្ញាតប៉ុណ្ណោះដែលអាចចូលប្រើEndpoint ដែលត្រូវបានការពារ។

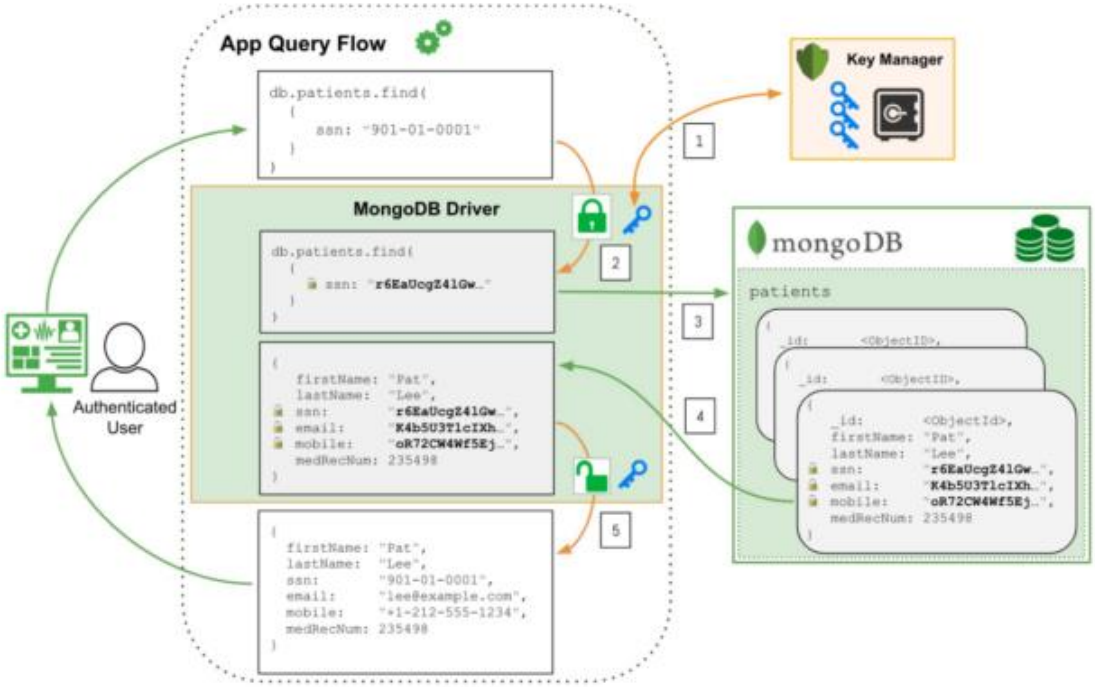
- **Input Validation:** ផ្នែកខាងក្រោយ **(Back-End)** ប្រើ validation libraries (eg. express-validator) ដើម្បីធានាថាទិន្នន័យ ដែលផ្តល់មកអោយប្រព័ន្ធត្រូវតាមទម្រង់ដែលរំពឹងទុក។ វាកាត់បន្ថយហានិភ័យនៃទិន្នន័យមិនប្រក្រតីដែលបណ្តាលឱ្យមានបញ្ហាណាមួយកើតឡើងនៅកំឡុងពេលប្រើប្រាស់ ។

- **Error Handling: API** ទាំងអស់ពេលដែលមាន បញ្ហាណាមួយកើតឡើងគឺ យើងត្រូវ តែបង្ហាញជាសារណាមួយដែល អាចអោយអ្នកប្រើប្រាស់បានដឹងពីបញ្ហាដែលកើតឡើងនៅពេលប្រើប្រាស់ ជាជាងយើងបង្ហាញព័ត៌មានដែលពាក់ព័ន្ធជានឹងព័ត៌មានសំខាន់ៗរបស់អ្នកប្រើប្រាស់។

៣.៤. ការផ្ទុកទិន្នន័យសុវត្ថិភាព (Secure Data Storage)

សុវត្ថិភាពនៃទិន្នន័យ ដែលរក្សាទុកក្នុងមូលដ្ឋានទិន្នន័យ (Data Base) គឺសំខាន់បំផុត។ ទិន្នន័យនៅក្នុង MongoDB ត្រូវតែត្រូវបានការពារពីការចូលប្រើ ការរំខាន ឬការបាត់បង់ដោយគ្មានការអនុញ្ញាត។ ការចូលប្រើមូលដ្ឋានទិន្នន័យ MongoDB ត្រូវបានដាក់កម្រិតដោយប្រើការត្រួតពិនិត្យការផ្ទៀងផ្ទាត់ និងការអនុញ្ញាត។ មានតែអ្នកប្រើប្រាស់ដែលមានការអនុញ្ញាតប៉ុណ្ណោះ ដែលអាចចូលប្រើមូលដ្ឋានទិន្នន័យ(Data Base) ដោយផ្ទាល់បាន។

ការពិចារណាអំពីសុវត្ថិភាព មានសារៈសំខាន់ខ្លាំងចំពោះភាពជោគជ័យនៃ ហាងអនឡាញ(Online Shop)។ តាមរយៈការអនុវត្ត ការផ្ទៀងផ្ទាត់ដ៏រឹងមាំ ការអ៊ិនគ្រីបទិន្នន័យ (Data Encryption) Secure APIs ប្រព័ន្ធធានាសុវត្ថិភាព និងឯកជនភាពរបស់អ្នកប្រើប្រាស់។ វិធានការទាំងនេះមិនត្រឹមតែការពារប្រព័ន្ធពីការគម្រាមកំហែងពីខាងក្រៅប៉ុណ្ណោះទេ ប៉ុន្តែថែមទាំងបង្កើតទំនុកចិត្តជាមួយអតិថិជន ដោយធានាថាទិន្នន័យរបស់ពួកគេត្រូវបានដោះស្រាយយ៉ាងយកចិត្តទុកដាក់បំផុត។



រូបភាពទី ៣.៤៖ បង្ហាញពី Secure Data Storage

ជំពូកទី ៤

បទពិសោធន៍អ្នកប្រើប្រាស់ និងការរចនា (User Experience and Design)

បទពិសោធន៍អ្នកប្រើប្រាស់ (UX) គឺជាទិដ្ឋភាពសំខាន់បំផុតមួយនៃការអភិវឌ្ឍន៍គេហទំព័រ ព្រោះវាប៉ះពាល់ដោយផ្ទាល់ពីរបៀបដែលអ្នកប្រើប្រាស់ធ្វើអន្តរកម្ម(Interactive)ជាមួយកម្មវិធី។ ជំពូកនេះបង្ហាញអំពីគោលការណ៍ នៃការរចនាដែលបានអនុវត្ត នៅក្នុងការអភិវឌ្ឍន៍ហាងអនឡាញ ដោយផ្តោតលើការរុករក (Navigation), responsive design និងសោភ័ណភាព។ វាក៏បង្ហាញពីរបៀបដែល Material UI (MUI) ត្រូវបានប្រើប្រាស់ ដើម្បីធានាបាននូវភាព consistency design ក្នុងប្រព័ន្ធទាំងមូល។

៤.១. គោលការណ៍រចនា (Design Principle)

- **ការរចនាផ្តោតលើអ្នកប្រើប្រាស់ (User-Centered Design) :** ការសម្រេចចិត្តរចនាទាំងអស់ត្រូវបានធ្វើឡើងដោយគិតគូរពីទស្សន:របស់អ្នកប្រើប្រាស់។ នេះរួមបញ្ចូលទាំងការរចនា ការរុករកដ៏សាមញ្ញ **(intuitive navigation)** និងការរចនាដែលមានភាពសាមញ្ញ ដើម្បីបង្កើត **responsive layout** ដែលដំណើរការយ៉ាងរលូនលើគ្រប់ទំហំឧបករណ៍ និងធានាថា Interface គ្មានភាពរញ៉េរញ៉ៃ និងងាយស្រួលយល់។

- **ភាពស៊ីសង្វាក់គ្នា (Consistency) : Consistency Design** គឺធ្វើអោយបទពិសោធន៍អ្នកប្រើប្រាស់ងាយស្រួល និងមានការចាប់អារម្មណ៍ មិនពិបាកក្នុងការប្រើប្រាស់។ ក្នុងប្រព័ន្ធមួយនេះបានយក MUI ដើម្បីជួយក្នុងការ **Design UI** ដូចជា **buttons, forms,** និង **typography** ដែលបានប្រកាន់ភ្ជាប់នូវ **Consistency Design** នៅទូទាំងកម្មវិធីទាំងមូល។

៤.២. Material UI Integration

Material UI (MUI) ត្រូវបានប្រើសម្រាប់ការរចនាហាងអនឡាញ។ MUI ផ្តល់នូវ wide range of pre-built components ដែលបានបង្កើតរួច ដែលមានរូបរាងស្អាត និងល្អប្រើសម្រាប់កម្មវិធីរបស់យើង។

- **បណ្តុំដែលអាចប្រើប្រាស់ឡើងវិញ (Component Reusability):** យើងអាចធ្វើអោយ MUI អាចប្រើឡើងវិញបានដែលកាត់បន្ថយ ពេលនៃការអភិវឌ្ឍន៍ និងផ្តល់នូវ consistent design នៅផ្នែកផ្សេងៗនៃកម្មវិធី ដូចជាបញ្ជីផលិតផល(product list), Cart Page និងCheckout Page។

- **ការកែលម្អ (Customization):** ខណៈពេលដែល MUI ផ្តល់នូវរចនាបទដែលមានលក្ខណៈស្រស់ស្អាត យើងអាចប្តូរតាមអ្វីដែលយើងចង់បាន បន្ថែមលើអ្វីដែលមានស្រាប់ដូចជាការ ផ្លាស់ប្តូរពណ៌ ការរាយអក្សរ

និងគម្លាតLayout ដើម្បីត្រូវជាមួយនឹងអ្វីដែលយើងចង់បាន ។

- Responsive Design: MUI's grid system មានសារៈសំខាន់ក្នុងការបង្កើត Responsive Layout

ជាមួយនឹង flexible breakpoints យើងបានធានាថាប្រព័ន្ធ ឬក៏កម្មវិធីរបស់យើង ដំណើរការបានយ៉ាងល្អនៅលើឧបករណ៍ចាប់ពីទូរសព្ទចល័តរហូតដល់អេក្រង់កុំព្យូទ័រ។

default breakpoints៖

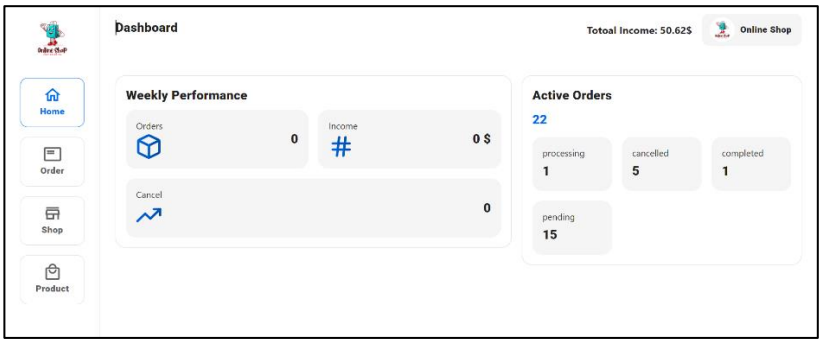
- xs, extra-small: 0px

- sm, small: 600px

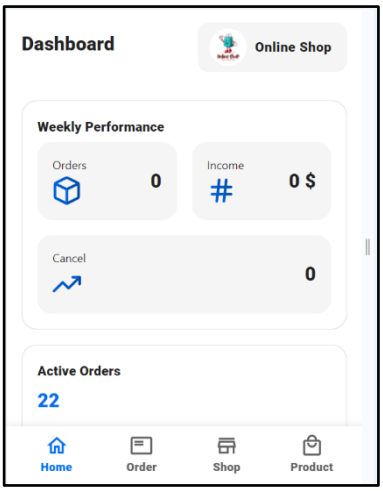
- md, medium: 900px

- lg, large: 1200px

- xl, extra-large: 1536px



រូបភាពទី ៤.២៖ បើកលើ Desktop



រូបភាពទី ៤.២៖ បើកលើ Mobile

៤.៣. User Interface (UI) Enhancements

នៅក្នុងចំណុចនេះនឹងបង្ហាញអំពីការកែលម្អមួយចំនួនដែលធ្វើការ Front-End មានភាពទាក់ទាញមានភាពងាយស្រួលក្នុងការប្រើប្រាស់៖

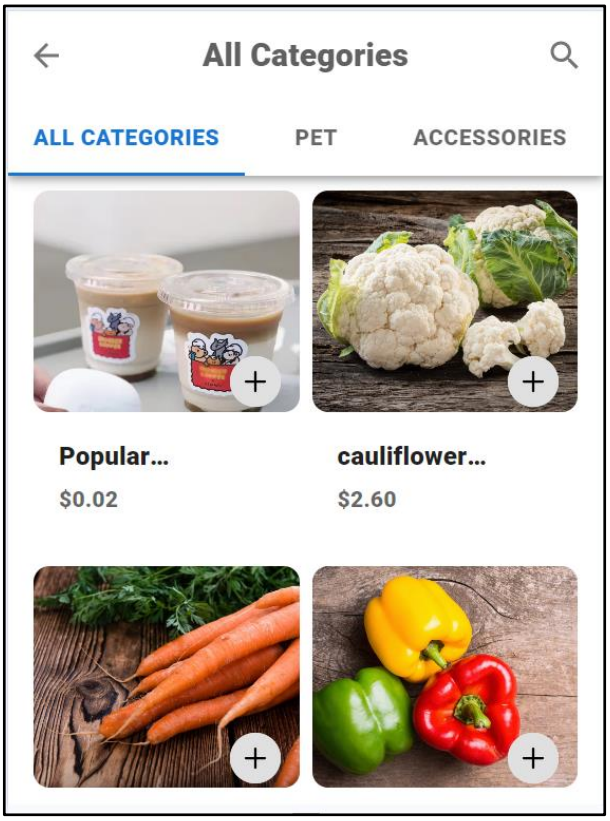
៤.៣.១. Navigation

ប្រើសម្រាប់អោយអ្នកប្រើប្រាស់ធ្វើការ ស្វែងរកផ្នែកណាមួយនៅក្នុងប្រព័ន្ធ របស់យើងមានភាពងាយស្រួលដោយបង្ហាញអ្នកប្រើប្រាស់ជាផ្នែក ដូចជា៖ product category, shopping cart, and account settings. Bottom menus and side navigation ត្រូវបានប្រើសម្រាប់ mobile ។

រូបភាពទី ៤.៣.១៖ បង្ហាញអំពី Navigation Bar

៤.៣.២. Product Listings

Product Listings ត្រូវបានរចនាឡើងជាមួយនឹងGrid Layout របស់ MUI ដើម្បីបង្ហាញរូបភាព ផលិតផល ឈ្មោះ និងតម្លៃ។ អ្នកប្រើប្រាស់អាចរកបានយ៉ាងងាយស្រួលតាមរយៈ Navigation និងមើលព័ត៌មានលម្អិតដោយចុចលើផលិតផល ។



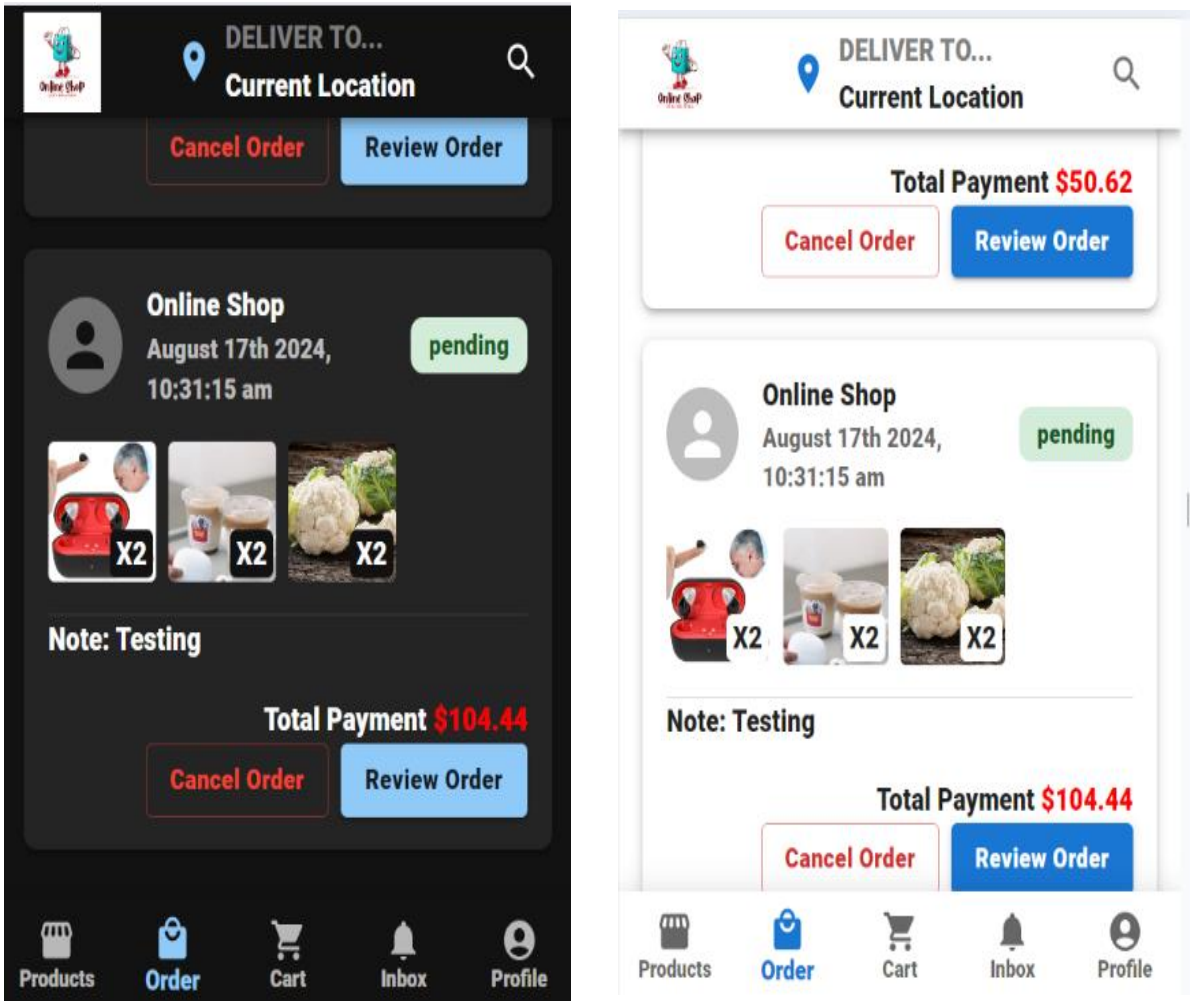
រូបភាពទី ៤.៣.២៖ បង្ហាញអំពី Product List

៤.៣.៣ Loading Progressive

ចំពោះ Loading Progressive គឺយើងគួរតែមាននៅក្នុងកម្មវិធីរបស់យើងដើម្បីធ្វើអោយ UX/UI របស់យើងកាន់តែប្រសើរឡើងសម្រាប់អ្នកប្រើប្រាស់ ភាគច្រើនយើងប្រើ Loading នៅពេលដែលយើងធ្វើការរងចាំការ response ពី backend តាមរយៈ api ធ្វើបែបនេះធ្វើអោយអ្នកប្រើប្រាស់ដឹងច្បាស់ថាយើងកំពុងតែរងចាំកិច្ចការណាមួយ។

៤.៣.៤. Dark Mode

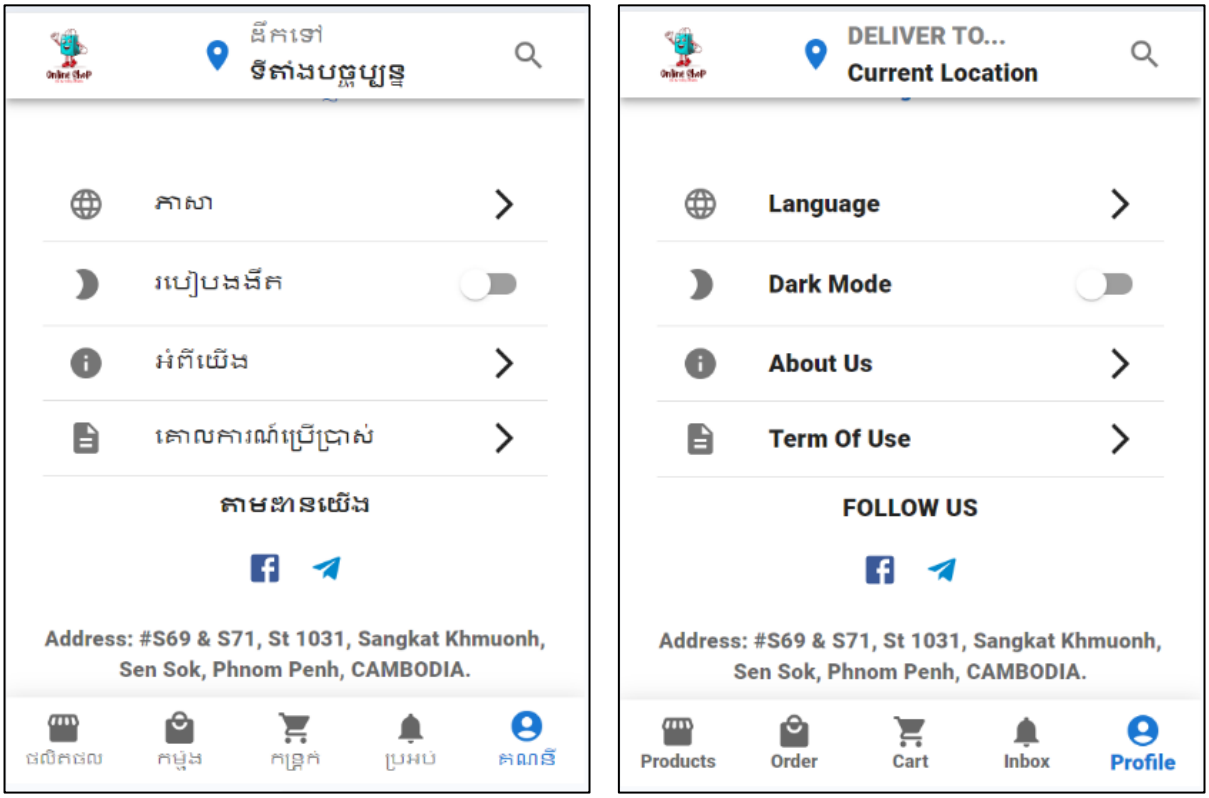
ផ្តល់អោយអ្នកប្រើប្រាស់នូវជម្រើសអោយអ្នកប្រើអាចប្រើកម្មវិធីក្នុងស្ថានភាពផ្សេងគ្នា **dark mode** សម្រាប់ផ្ទៃងងឹត និង **light mode** សម្រាប់ផ្ទៃភ្លឺ ។



រូបភាពទី ៤.៣.៤៖ បង្ហាញអំពី Dark Mode & Light

៤.៣.៥ ការផ្លាស់ប្តូរភាសា (Change Language)

ការផ្លាស់ប្តូរភាសាមានសារៈសំខាន់មួយផ្នែកទៀតផងដែរសម្រាប់អោយអ្នកប្រើប្រាស់ងាយស្រួលក្នុងការប្រាស់ដោយសារតែអ្នកប្រើប្រាស់មួយចំនួនអាចជា ជនជាតិបរទេស ខ្លះជាជនជាតិខ្មែរ ដូច្នេះការផ្លាស់ប្តូរភាសាធ្វើអោយការប្រើប្រាស់ប្រព័ន្ធរបស់យើងគឺកាន់តែមានការងាយស្រួល។



រូបភាពទី ៤.៣.៥៖ បង្ហាញអំពីការផ្លាស់ប្តូរភាសា

ការរចនានៃហាងអនឡាញ ត្រូវតែផ្តោតទៅលើការផ្តល់នូវបទពិសោធន៍អ្នកប្រើប្រាស់អោយមានការទាក់ទាញ និងងាយស្រួលក្នុងការប្រើប្រាស់។ ដោយប្រើ consistent design និង responsive design ដែលអាចអោយអ្នកប្រើប្រាស់ចូលប្រើបានគ្រប់ប្រភេទ Device ទាំងអស់ ។

ជំពូកទី៥

លទ្ធផលនៃការស្រាវជ្រាវ

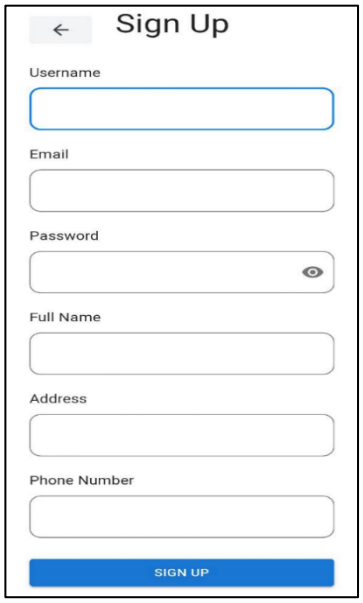
៥.១. និដ្ឋភាពទូទៅ

ជំពូកនេះបង្ហាញពីលទ្ធផលនៃការអភិវឌ្ឍហាងអនឡាញ ដោយប្រើបច្ចេកវិទ្យាអភិវឌ្ឍន៍គេហទំព័រទំនើបដូចជា React.js, Node.js, Express.js, MongoDB និង Mongoose ។ ជំពូកនេះនឹងគ្របដណ្តប់លើមុខងារនៃប្រព័ន្ធដែលបានអនុវត្ត ។ គោលដៅគឺដើម្បីវាយតម្លៃប្រសិទ្ធភាពនៃដំណោះស្រាយ និងថាតើវាឆ្លើយតបទៅនឹងគោលបំណងគម្រោងដែលបានរៀបរាប់ពីមុននៅក្នុងសារណានេះកម្រិតណា។

៥.២. លទ្ធផល

៥.២.១. ការចុះឈ្មោះអ្នកប្រើប្រាស់ និង Authentication

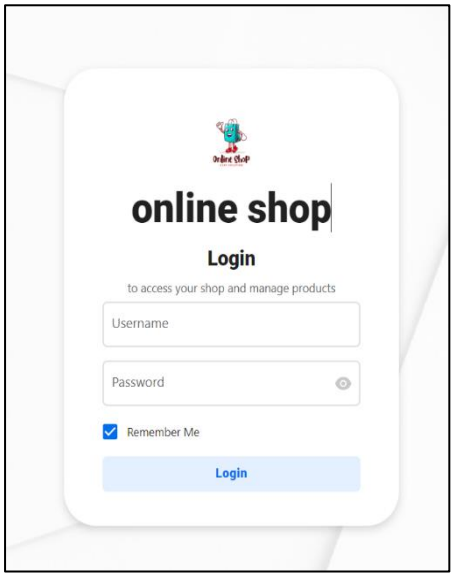
កម្មវិធីនេះអនុញ្ញាតឱ្យអ្នកប្រើប្រាស់ចុះឈ្មោះដោយប្រើឈ្មោះអ្នកប្រើប្រាស់ អ៊ីម៉ែល និងពាក្យសម្ងាត់ ទីតាំងអ្នកប្រើប្រាស់ លេខទូរស័ព្ទ ក្នុងការបង្កើតគណនីសម្រាប់ប្រើប្រាស់នៅក្នុងកម្មវិធី រួមទាំង Encryption Password ផងដែរពេលដែលអ្នកប្រើប្រាស់បង្កើតគណនីរួចរាល់ ការដែលយើង Encrypt Password នេះគឺដើម្បីធ្វើអោយអ្នកប្រើប្រាស់មានភាពជឿទុកចិត្តលើប្រព័ន្ធរបស់យើងបន្ថែម។



រូបភាពទី ៥.២.១ ៖ បង្ហាញអំពី Register Response Backend

រូបភាពទី ៥.២.១ ៖ បង្ហាញអំពីការចុះឈ្មោះអ្នកប្រើប្រាស់

- បានអនុវត្ត JWT (JSON Web Token) សម្រាប់ការផ្ទៀងផ្ទាត់អ្នកប្រើប្រាស់ប្រកបដោយសុវត្ថិភាព ដោយផ្តល់ឱ្យអ្នកប្រើប្រាស់នូវសញ្ញាសម្ងាត់ដែលត្រូវការសម្រាប់ការចូលប្រើប្រាស់កម្មវិធី។

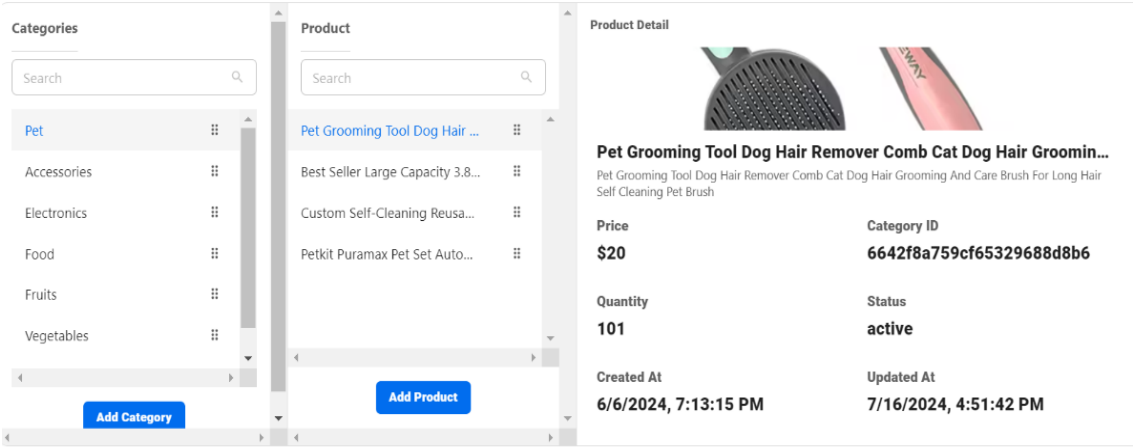


រូបភាពទី ៥.២.១ ៖ បង្ហាញអំពីការ Login

រូបភាពទី ៥.២.១ ៖ បង្ហាញអំពីការ Login Response Backend

៥.២.២. ការគ្រប់ផលិតផល

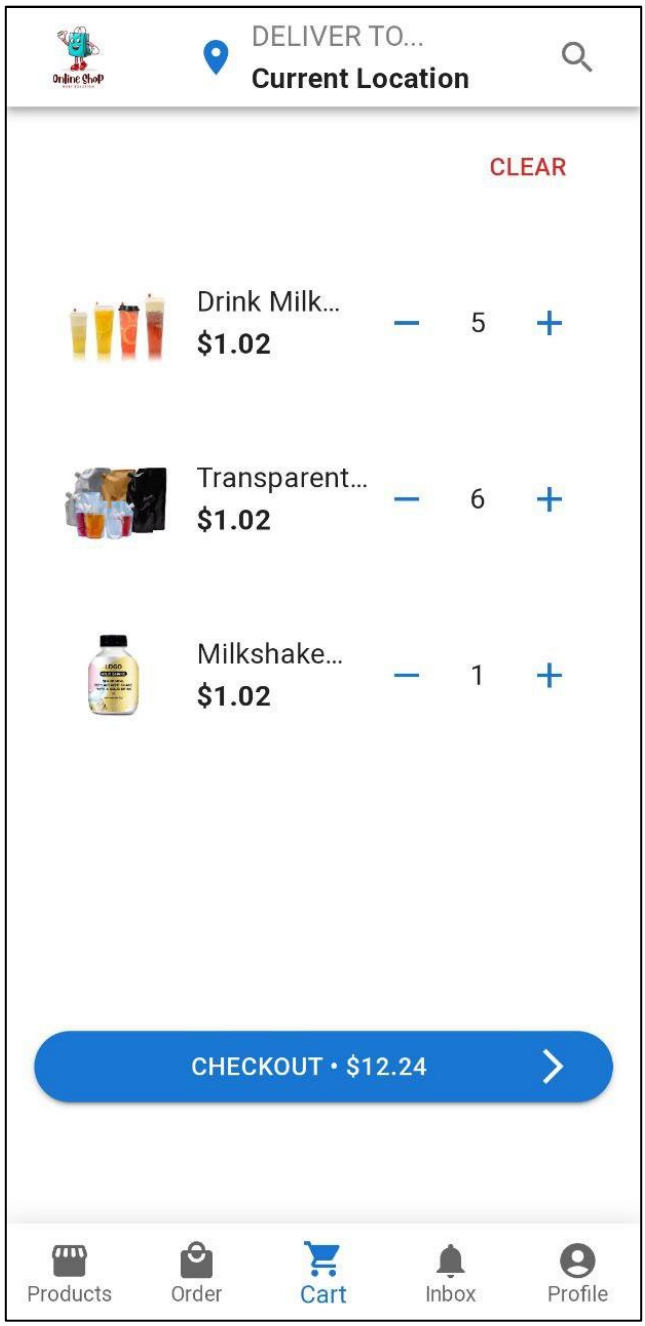
ប្រព័ន្ធផ្តល់នូវ interface សម្រាប់អ្នកគ្រប់គ្រងហាងអាច បន្ថែម ធ្វើបច្ចុប្បន្នភាព លុប ស្វែងរក ផលិតផល និង ប្រភេទផលិតផល និងការគ្រប់គ្រងផលិតផល។ បានអនុវត្តប្រតិបត្តិការ CRUD ដោយប្រើ Node.js និង MongoDB ដែលផលិតផលត្រូវបានរក្សាទុកនៅក្នុង MongoDB Database និងត្រូវបានគ្រប់គ្រងតាមរយៈ Mongoose Schema ។ ចំណែកឯផ្នែកខាងមុខ (React.js) បង្ហាញទិន្នន័យផលិតផលជាលក្ខណៈdynamically រួមទាំងរូបភាព ការពិពណ៌នា និងតម្លៃ ដែលអនុញ្ញាតឱ្យអ្នកប្រើប្រាស់រុករក ប្រកបដោយប្រសិទ្ធភាព។



រូបភាពទី ៥.២.២ ៖ បង្ហាញអំពីការគ្រប់គ្រងផលិតផល

៥.២.៣. Shopping Cart

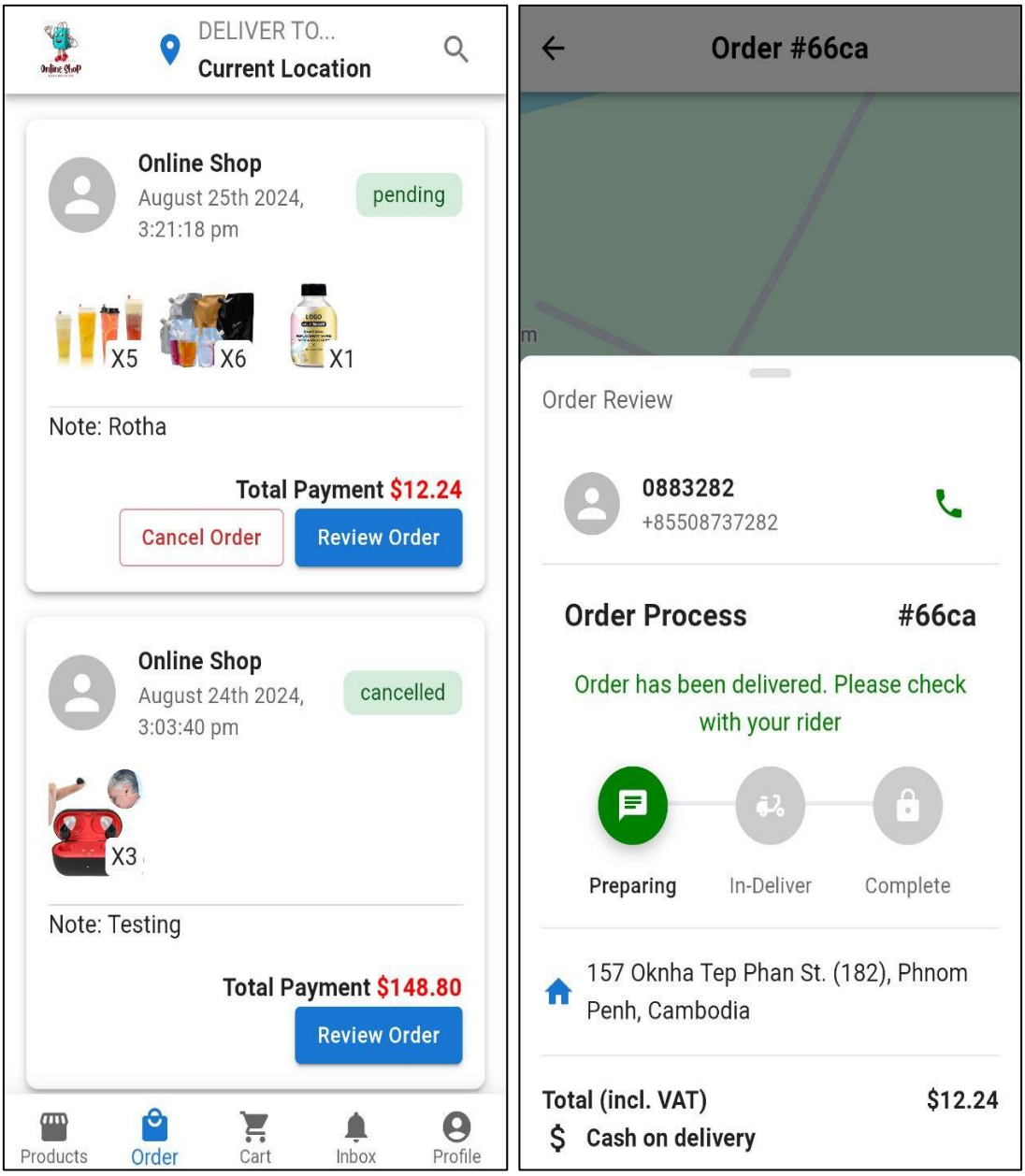
បានបង្កើតដើម្បីឆ្លើយតបទៅអ្នកប្រើប្រាស់ដែលអាចបន្ថែម ធ្វើបច្ចុប្បន្នភាព ឬលុប items។ អនុវត្ត Real-Time Updates ដោយប្រើការគ្រប់គ្រង React State និង Data Flow ដែលធ្វើអោយការទាក់ទងប្រសើរឡើងរវាងសមាសធាតុ ដើម្បីធានាបាននូវការប្រើប្រាស់ដោយរលូន។ **Shopping Cart** ត្រូវបានរក្សាទុកនៅក្នុង **Local Storage** ដើម្បីផ្តល់នូវបទពិសោធន៍ការទិញទំនិញកាន់តែមានភាពល្អប្រសើរ។



រូបភាពទី ៥.២.៣ ៖ បង្ហាញអំពីផលិតផលដាក់ចូលក្នុង Cart

៥.២.៤. Order List

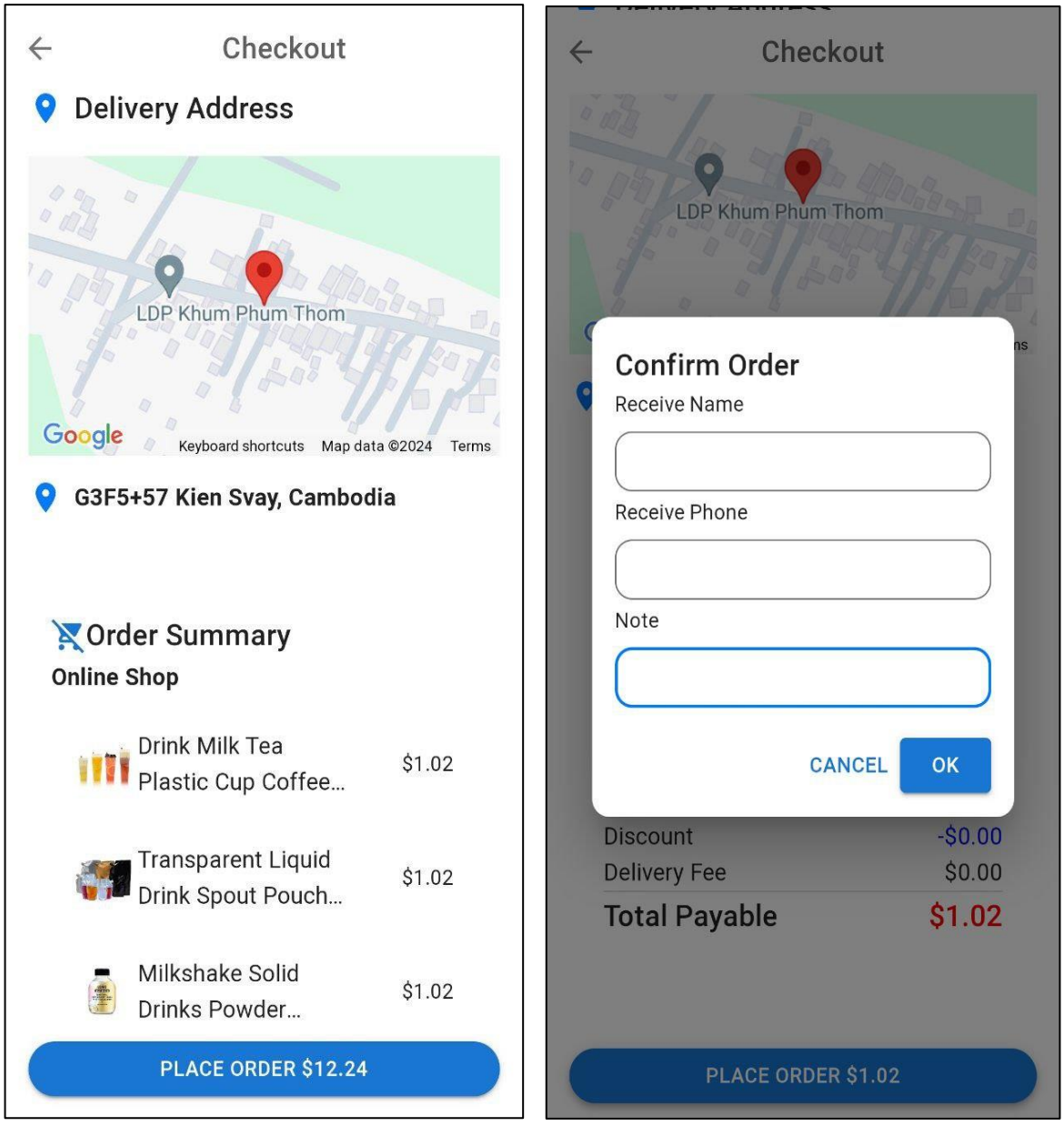
ក្នុងផ្នែក Order List អ្នកប្រើប្រាស់អាចមើលទៅលើ List Order ដែលគាត់បានធ្វើការ Order ដែលក្នុងនោះអ្នកប្រើប្រាស់អាចមើល Order Detail អាចធ្វើការ Cancelled Order នៅពេលដែល Order Status ស្មើនឹង Pending តែអ្នកប្រើប្រាស់មិនអាចធ្វើការ Cancel Order បានទេប្រសិនជា Order Status ស្មើនឹង Processing ។



រូបភាពទី ៥.២.៤ ៖ បង្ហាញអំពី Order List និង Order Detail

៥.៣. Order Processing and Checkout

នៅក្នុងការកម្ម៉ង់ផលិតផលអ្នកប្រើប្រាស់អាចជ្រើរើសផលិតផលណាដែលពេញចិត្តដាក់ចូលទៅក្នុង Cart ទើបអាចធ្វើការ កម្ម៉ង់ផលិតផលបាន នៅកំឡុងពេលដែលធ្វើការ Order ប្រព័ន្ធរបស់យើងនឹងចាប់បានទីតាំងជាក់លាក់មួយរបស់អ្នកប្រើប្រាស់ដើម្បីដឹងពីទីតាំងរបស់អ្នកប្រើប្រាស់ពេលដែលធ្វើការ Order អ៊ីវ៉ាន់ ។ បន្ទាប់មកពេលដែលអ្នកប្រើប្រាស់ធ្វើការ Order នឹងតម្រូវអោយអ្នកប្រើប្រាស់បំពេញព័ត៌មានបន្ថែមមួយចំនួន ដូចជា ឈ្មោះ លេខទូរសព្ទ និង Note ដើម្បីធ្វើការ Order ។

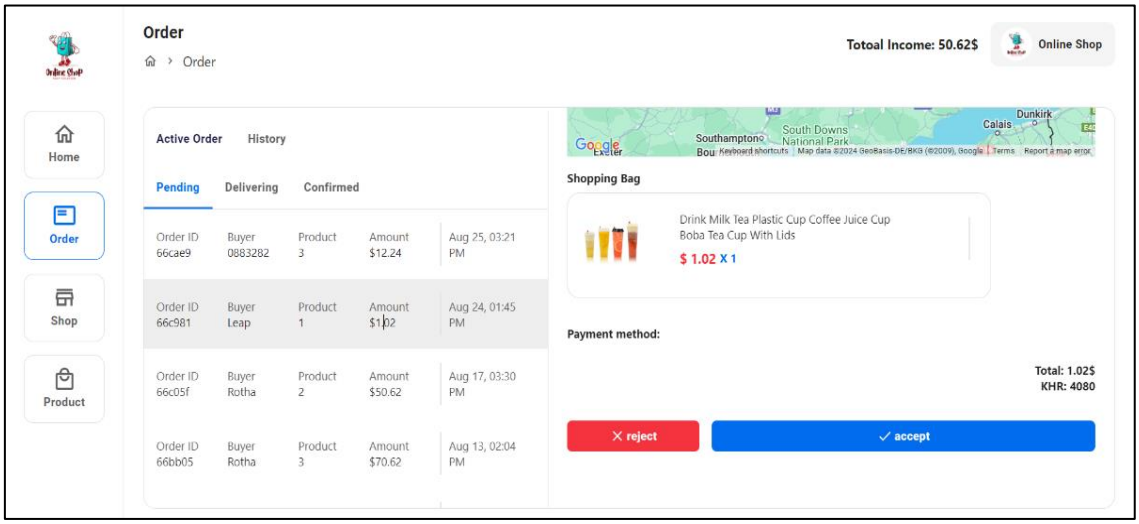


រូបភាពទី ៥.៣ ៖ បង្ហាញអំពីការ Order Processing and Checkout

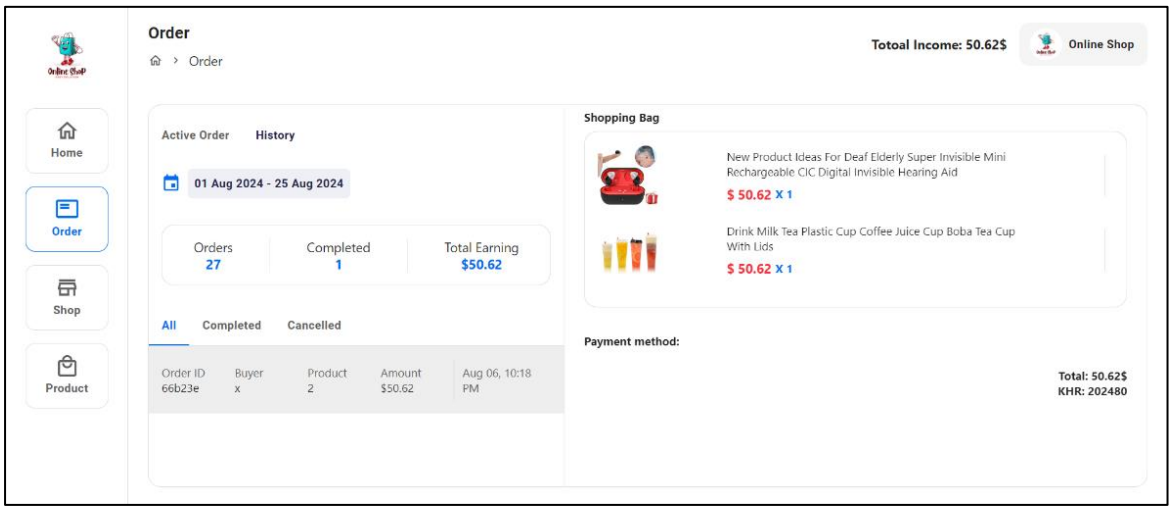
៥.៣.១. Admin Dashboard

Admin Dashboard ត្រូវបានបង្កើតឡើងដើម្បីតាមដានសកម្មភាពអ្នកប្រើប្រាស់ ការលក់ និងគ្រប់គ្រងផលិតផលក្នុងហាង។ បានប្រើប្រាស់បច្ចេកទេសប្រមូលផ្តុំទិន្នន័យនៅក្នុង MongoDB ដើម្បីផ្តល់ការយល់ដឹងអំពីឥរិយាបថអ្នកប្រើប្រាស់ និងដឹងពីលំហូរទិន្នន័យ។ **Admin Dashboard** ក៏រួមបញ្ចូលការមើលឃើញផងដែរ ដូចជាចំនួននៃការ order របស់អ្នកប្រើប្រាស់ការ Cancelled Orderចំនួនថវិការដែលរកបានក្នុង១ថ្ងៃនិងថវិការសរុបដែលហាងរកបានថ្ងៃកន្លងៗទៅ លើសពីនេះទៅទៀត **Admin Dashboard** អាចប្រើក្នុងការយល់ព្រមការកម្ម៉ង់ ឬ បោះបង់បាន និងការគ្រប់គ្រង ផលិតផល។

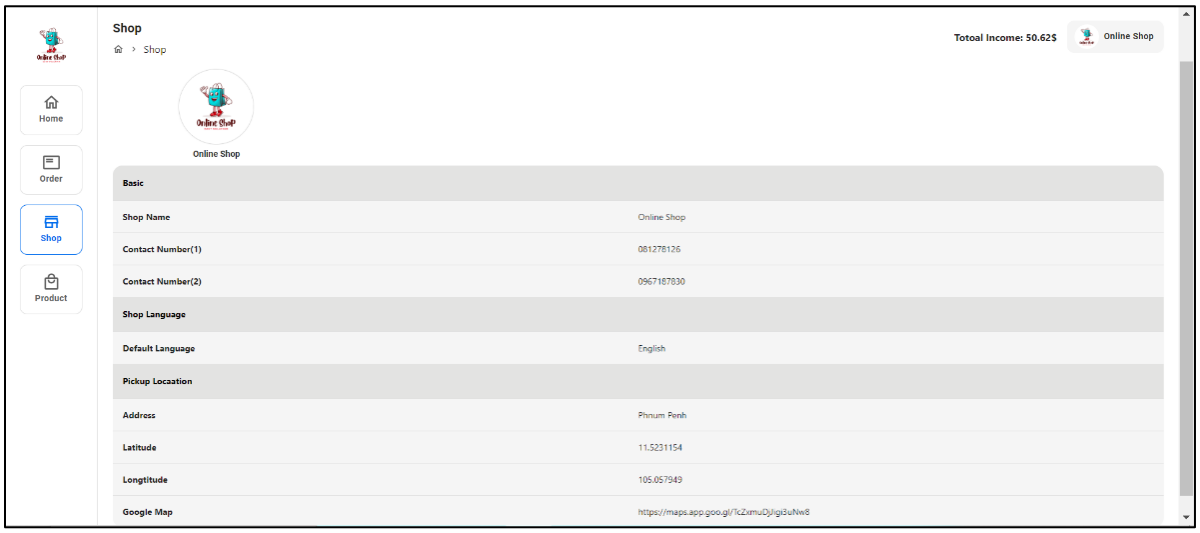
រូបភាពទី ៥.៣.១ ៖ បង្ហាញអំពីការ Dashboard



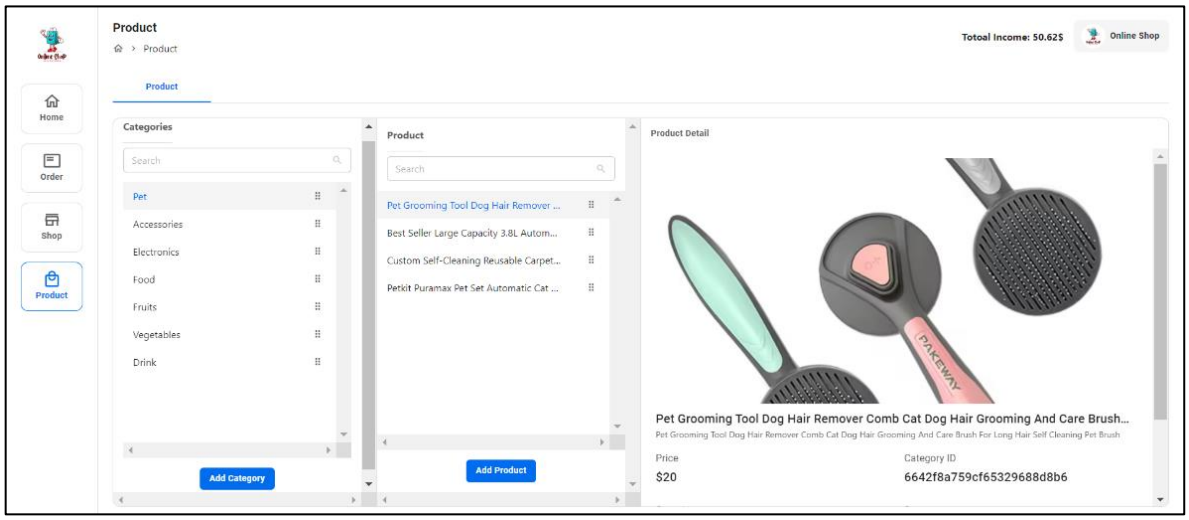
រូបភាពទី ៥.៣.១ ៖ បង្ហាញអំពីការ Active Order



រូបភាពទី ៥.៣.១ ៖ បង្ហាញអំពីការ Order History



រូបភាពទី ៥.៣.១ ៖ បង្ហាញអំពីការ Shop Detail



រូបភាពទី ៥.៣.១ ៖ បង្ហាញអំពីការ Product Management

៥.៣.២. កាត់ Stock

ពេលដែល User ធ្វើការ Order ផលិតផលដែលមាននៅក្នុង Stock នឹងត្រូវបានដកចេញតាមចំនួន Order របស់អ្នកប្រើប្រាស់ដើម្បីមានភាពងាយស្រួលដល់ម្ចាស់អាជីវកម្មក្នុងការគ្រប់គ្រងផលិតផលរបស់គាត់។

៥.៤. សន្និដ្ឋានលទ្ធផល

កម្មវិធីហាងអនឡាញ ដែលបានបង្កើតដោយជោគជ័យបំពេញតាមគោលបំណងគម្រោងក្នុងការផ្តល់នូវវេទិកាដែលមាន ភាពងាយស្រួលប្រើ។ ការរួមបញ្ចូលគ្នានៃបច្ចេកវិជ្ជា និងការរចនា ដែលអាចបម្រើទាំងអាជីវកម្មខ្នាតតូច និងមធ្យមប្រកបដោយប្រសិទ្ធភាព។ ការងារនាពេលអនាគតនឹងផ្តោតលើការបង្កើនបទពិសោធន៍អ្នកប្រើប្រាស់ រួមបញ្ចូលការវិភាគកម្រិតខ្ពស់ និងការពង្រីក លក្ខណៈពិសេសដោយផ្អែកលើមតិអ្នកប្រើ។

សេចក្តីសន្និដ្ឋាន និងការផ្តល់អនុសាសន៍

ចុងក្រោយនេះ យើងសង្ខេបរកឃើញចំណុចសំខាន់ៗ និងការរួមចំណែកនៃគម្រោង។ ការស្រាវជ្រាវបានផ្តោតលើការបង្កើតហាងអនឡាញដែលមានសុវត្ថិភាព និង អាចធ្វើការអភិវឌ្ឍន៍ បន្តបន្ថែមបានដោយប្រើបច្ចេកវិទ្យាមួយចំនួនដូចជា React.js, Node.js, Express.js និង MongoDB ។ ការអនុវត្តបានផ្តល់នូវការយល់ដឹងដ៏មានតម្លៃចំពោះការអភិវឌ្ឍន៍កម្មវិធីគេហទំព័រ សុវត្ថិភាព និងscalability ។

៦.១. សេចក្តីសន្និដ្ឋាន

ការបញ្ចប់គម្រោងនេះ គឺជាសមិទ្ធិផលដ៏សំខាន់មួយនៅក្នុងវិស័យអភិវឌ្ឍន៍គេហទំព័រ ជាពិសេសនៅក្នុងបរិបទនៃពាណិជ្ជកម្មអេឡិចត្រូនិក។ ហាងអនឡាញរួមបញ្ចូលគ្នាដោយជោគជ័យនូវបច្ចេកវិទ្យាគេហទំព័រទំនើបដើម្បីបង្កើតវេទិកាដែលមានមុខងារ សុវត្ថិភាព និងងាយស្រួលប្រើ។ ទោះជាយ៉ាងណាក៏ដោយ ដែនកំណត់របស់គម្រោងក៏បង្ហាញពីតម្រូវការសម្រាប់ការអភិវឌ្ឍន៍ និងការកែលម្អជាបន្ត។

ដោយសារទិដ្ឋភាពឌីជីថលបន្តវិវឌ្ឍ ដូច្នេះហើយ ហាងអនឡាញក៏ត្រូវតែមានការអភិវឌ្ឍន៍បន្តទៀតផងដែរ។ ការងារនាពេលអនាគត ដែលបានរៀបរាប់នៅក្នុងជំពូកនេះផ្តល់នូវគំនិតបង្ហាញផ្លូវ សម្រាប់ការកែលម្អបន្ថែមទៀត ដោយធានាថាប្រព័ន្ធនេះនៅតែមានការប្រកួតប្រជែង សុវត្ថិភាព និងមានសមត្ថភាពបំពេញតម្រូវការរបស់អ្នកប្រើប្រាស់របស់ខ្លួន។

ដំណើរការនៃការអភិវឌ្ឍន៍នៃហាងអនឡាញនេះមិនទាន់បញ្ចប់នៅត្រឹមនេះទេ។ ផ្ទុយទៅវិញ វាគឺជាការចាប់ផ្តើមនៃដំណើរការបន្តនៃការកែលម្អ និងការច្នៃប្រឌិត។ តាមរយៈការដោះស្រាយ limitations និងការស្វែងរកឱកាសសម្រាប់ការងារនាពេលអនាគត ហាងអនឡាញនិងអាចវិវឌ្ឍទៅជាវេទិកាដ៏ទូលំទូលាយ និងសម្បូរបែបដែលបម្រើដល់អ្នកប្រើប្រាស់ចម្រុះ នៅក្នុងជាសកល។

៦.២. ការផ្តល់អនុសាសន៍

មានរបស់មួយចំនួនសម្រាប់ការអភិវឌ្ឍន៍ នាពេលអនាគតអាចបង្កើនមុខងារ សុវត្ថិភាព និងបទពិសោធន៍អ្នកប្រើប្រាស់នៃហាងអនឡាញ ៖

- មុខងារសុវត្ថិភាព (Enhanced Security Features) ៖ ការងារនាពេលអនាគតគួរតែផ្តោតលើការអនុ

វត្តវិធានការសុវត្ថិភាពកម្រិតខ្ពស់ដូចជាការផ្ទៀងផ្ទាត់ two-factor, biometric login options និងបច្ចេកទេស អ៊ិនគ្រីបដ៏រឹងមាំបន្ថែមទៀត។

- **Performance Optimization** ៖ នៅពេលដែលអ្នកប្រើប្រាស់កើនឡើង ការបង្កើនប្រសិទ្ធភាពប្រតិបត្តិការ នឹងកាន់តែមានសារៈសំខាន់។ បច្ចេកទេសដូចជា database indexing, query optimization, and content delivery networks (CDNs) អាចត្រូវបានរុករកដើម្បីធានាបាននូវ fast load times និងបទពិសោធន៍អ្នកប្រើប្រាស់យ៉ាងរលូន។

- **Machine Learning Integration** ៖ ការរួមបញ្ចូលក្បួនដោះស្រាយ Machine Learning អាចជួយបង្កើនបទពិសោធន៍អ្នកប្រើប្រាស់យ៉ាងសំខាន់។ ជាឧទាហរណ៍ ការបង្កើតប្រព័ន្ធណែនាំអ្នកប្រើប្រាស់ដោយផ្អែកលើអាកប្បកិរិយារបស់អ្នកប្រើប្រាស់ និងប្រវត្តិការទិញអាចកំណត់បទពិសោធន៍ទិញទំនិញផ្ទាល់ខ្លួន ដែលនាំឱ្យមានការបង្កើនការប្រើប្រាស់កម្មវិធី ឬប្រព័ន្ធ និងការលក់។

- **Internationalization and Localization**៖ ការពង្រីកប្រព័ន្ធដើម្បីប្រើច្រើនភាសា និងរូបិយប័ណ្ណជាច្រើននឹងធ្វើឱ្យវាអាចចូលប្រើបានសម្រាប់ ប្រជាជនទូទាំងពិភពលោក។ វានឹងពាក់ព័ន្ធនឹងការកែសម្រួល Interface សម្រាប់អ្នកប្រើប្រាស់ content និងតម្លៃ តាមតំបន់ និងវប្បធម៌ផ្សេងៗ។

- **Mobile Application Development**៖ ខណៈពេលបច្ចុប្បន្នកម្មវិធីរបស់យើងមានការ responsive ហើយ តែការអភិវឌ្ឍន៍កម្មវិធីទូរស័ព្ទ អាចផ្តល់នូវបទពិសោធន៍អ្នកប្រើប្រាស់កាន់តែល្អមួយកម្រិតទៀតសម្រាប់អ្នកប្រើប្រាស់ដែលចូលចិត្តប្រើApp ជាង Web ។ វានឹងពាក់ព័ន្ធនឹងការបង្កើតកម្មវិធី សម្រាប់ប្រព័ន្ធប្រតិបត្តិការ iOS និង Android ដោយប្រើប្រាស់ API ផ្នែកខាងក្រោយដែលមានស្រាប់។

- **Advanced Analytics and Reporting** ៖ការអនុវត្តឧបករណ៍វិភាគកម្រិតខ្ពស់អាចផ្តល់នូវការយល់ដឹងដ៏មានតម្លៃចំពោះអាកប្បកិរិយាអ្នកប្រើប្រាស់ និន្នាការលក់។ ការយល់ដឹងទាំងនេះអាចត្រូវបានប្រើដើម្បីធ្វើការសម្រេចចិត្តដែលផ្អែកលើទិន្នន័យ ធ្វើអោយប្រសើរឡើងនូវប្រសិទ្ធភាពទាំងមូលនៃហាងអនឡាញ។

- **Voice Search and AI Assistants**៖ ការរួមបញ្ចូលសមត្ថភាពស្វែងរកដោយប្រើសំឡេងនិងជំនួយការដែលដំណើរការដោយ AI អាចបង្កើនភាពងាយស្រួល និងលទ្ធភាពប្រើប្រាស់របស់ហាងមួយកម្រិតទៀត។ ការដំណើរការដោយសំឡេងនឹងអនុញ្ញាតឱ្យអ្នកប្រើប្រាស់ស្វែងរកផលិតផល ដោយមិនចាំបាច់បញ្ចូលដោយដៃ ដែលធ្វើឱ្យបទពិសោធន៍ទិញទំនិញកាន់តែងាយស្រួល។

- **Dynamic Pricing Algorithms**៖ ការបង្កើត dynamic pricing algorithms ដែលអាចធ្វើការកែតម្រូវតម្លៃដោយផ្អែកលើតម្រូវការ ផលិតផលដែលមាននៅក្នុង Stock និងតម្លៃជាមួយគូប្រជែងអាចបង្កើនប្រាក់ចំណូលបន្ថែម។ វិធីសាស្ត្រនេះនឹងរួមបញ្ចូលការវិភាគទិន្នន័យចំពោះពេលវេលាជាក់ស្តែង និងចំណូលបន្ថែម។

វិធីសាស្ត្រនេះនឹងរួមបញ្ចូលការវិភាគទិន្នន័យចំពោះពេលវេលាជាក់ស្តែង និង machine learning to predict trends និងកំណត់តម្លៃដ៏ល្អប្រសើរមួយសម្រាប់ផលិតផល។

ឯកសារយោង

ឯកសារយោងតាមសៀវភៅមានដូចខាងក្រោម៖

១. Node.js Design Patterns: by Mario Casciaro, Luciano Mammino

សៀវភៅនេះបានលើកឡើងពី architecture and design patterns ដែលត្រូវបានធ្វើដោយប្រើ Node.js ដែលបានបង្ហាញនៅក្នុងផ្នែក system architecture នៃសារណា។

២. MongoDB: The Definitive Guide: by Shannon Bradshaw, Eoin Brazil, Kristina Chodorow

សៀវភៅនេះបានផ្តល់នូវការយល់ដឹងយ៉ាងស៊ីជម្រៅនៃ MongoDB ដែលបានប្រើនៅក្នុង database architecture ក្នុង online shop project នេះ។

៣. JavaScript The Good Parts: by Douglas Crockford

សៀវភៅនេះបានពន្យល់ច្បាស់ពី JavaScript ដែលអាចរួមបញ្ចូលទាំងនៅក្នុង React.js និង Node.js ក្នុងការបង្កើតសារណានេះ។

៤. React Up & Running: Building Web Applications: by Stoyan Stefanov

សៀវភៅនេះបានផ្តល់យើងនូវការអនុវត្តលើការបង្កើតគេហទំព័រជាមួយ Reactjs ដែលយើងបានយកមកប្រើសម្រាប់ frontend របស់យើង ។

៥. Express in Action: Writing, Building, and Testing Node.js Applications: by Evan Hahn

សៀវភៅនេះបានផ្តោតទៅលើ Express.js នៅក្នុង Node.js ដែលពាក់ព័ន្ធជាមួយការបង្កើត API នៅក្នុងសារណាមួយនេះ។

៦. Material-UI: The Complete Guide: by Karl Swedberg

សៀវភៅនេះពន្យល់លំអិតនៃការប្រើ Material-UI សម្រាប់ React components ដែលយើងបានធ្វើនៅលើ Frontend នៃហាងអនឡាញរបស់យើង។

ឯកសារយោងតាមគេហទំព័រមានដូចខាងក្រោម៖

១. Mozilla Developer Network (MDN Web Docs): - https://developer.mozilla.org/

នៅក្នុងគេហទំព័រមួយនេះបានផ្តល់ចំណេះដឹងជាច្រើនទៅនឹង JavaScript, HTML, CSS, និងលក្ខណៈទូទៅនៃការអភិវឌ្ឍន៍គេហទំព័រ ដែលបានឆ្លុះបញ្ចាំងតាមរយៈការធ្វើនៅក្នុង Online Shop ។

២. Node.js Official Documentation: - https://nodejs.org/en/docs/

នៅក្នុងគេហទំព័រមួយនេះបានផ្តល់ចំណេះដឹងអំពី Node.js features, best practices និងការប្រើ API នៅក្នុង Backend ។

៣. MongoDB Documentation: - https://docs.mongodb.com/

នៅក្នុងគេហទំព័រមួយនេះបានផ្តល់ចំណេះដឹងអំពីការបង្កើត MongoDB databases, ការគ្រប់គ្រងទិន្នន័យ ដែលបានបង្ហាញនៅក្នុងជំពូក database architecture ។

៤. React.js Official Documentation: - https://react.dev/learn

នៅក្នុងគេហទំព័រមួយនេះបានផ្តល់ចំណេះដឹងអំពីពាក់ព័ន្ធទៅនឹង React.js ដែលយើងបានធ្វើនៅក្នុងជំពូក frontend development ។

៥. Express.js Documentation: - https://expressjs.com/

នៅក្នុងគេហទំព័រមួយនេះបានផ្តល់ចំណេះដឹងអំពីពាក់ព័ន្ធទៅនឹង Express.js សម្រាប់បង្កើត APIs ដែលជាផ្នែកសំខាន់នៃការរចនាប្រព័ន្ធ។

៦. Material-UI Documentation: - https://mui.com/

នៅក្នុងគេហទំព័រមួយនេះបានផ្តល់ចំណេះដឹងអំពីពាក់ព័ន្ធទៅនឹង Material-UI components ។

៧. OWASP Foundation: - https://owasp.org/

នៅក្នុងគេហទំព័រមួយនេះបានផ្តល់ចំណេះដឹងអំពីពាក់ព័ន្ធទៅនឹង web application security ដែលបានបង្ហាញនៅក្នុងជំពូក ការពិចារណាអំពីសុវត្ថិភាព (Security Considerations) ។

៧. W3C Web Security: - https://www.w3.org/Security/

នៅក្នុងគេហទំព័រមួយនេះបានផ្តល់ចំណេះដឹងអំពីពាក់ព័ន្ធទៅនឹង ស្តង់ដារសុវត្ថិភាពគេហទំព័រ យោងនៅក្នុងការពិចារណាអំពីសុវត្ថិភាព (Security Considerations) ។

៨. Stack Overflow: - https://stackoverflow.com/

ប្រើញឹកញាប់សម្រាប់ការដោះស្រាយបញ្ហា និងការស្វែងរកដំណោះស្រាយអំឡុងពេលអភិវឌ្ឍន៍ ។

ឧបសម្ព័ន្ធ

1- Backend

1.1. Define Server

```
require("dotenv").config( );

const express = require("express");

const mongoose = require("mongoose");

const app = express( );

const path = require("path");

const PORT = process.env.PORT;

const URL = process.env.MOGO_URL;

var cors = require("cors");

app.use(cors( ));
```



```
app.listen(PORT, ( ) => {

  console.log(`Server is running on http://localhost:${PORT}`);

});

app.get("/", (req, res) => {

  res.send("Hello, world!");
```

```
});
```

telegram.js

```
async function sendMessage(chatId, text) {
  const botToken = "6823978357:AAHZn1uoJ6W__yqEghnsaPql7BFG7cveDgU";
  const apiUrl = `https://api.telegram.org/bot${botToken}/sendMessage`;
  try {
    await axios.post(apiUrl, {
      chat_id: chatId,
      text,
    });
  } catch (error) {
    console.error("Error sending Telegram message:", error.message);
  }
}
```

multer.js

```
const multer = require("multer");
const storage = multer.memoryStorage();
const upload = multer({ storage: storage });
module.exports = upload;
```

1.2. Define Route

1.2.1. auth.js

```
const express = require("express");

const router = express.Router();

const authController = require("../controllers/authController");

const resetPasswordController = require("../controllers/resetPasswordController");

router.post("/signup", authController.signup);

router.post("/login", authController.login);

router.post("/reset-password", resetPasswordController.resetPassword);

module.exports = router;
```

1.2.2. notification.js

```
const express = require("express");

const router = express.Router();

const notificationController = require("../controllers/notificationController");

router.post("/", notificationController.createNotification);

router.get("/", notificationController.getNotifications);
```

```
router.put( "/:id/read", notificationController.updateNotificationReadStatus );

module.exports = router;
```

1.2.3. orderroute.js

```
const express = require( "express" );

const orderController = require( "../controllers/orderController" );

const router = express.Router( );

router.post( "/", orderController.createOrder );

router.get( "/user/:userId", orderController.getUserOrders );

router.get( "/", orderController.getOrders );

router.get( "/total-sales", orderController.getTotalSales );

router.get( "/:id", orderController.getOrderById );

router.patch( "/:id/status", orderController.updateOrderStatus );

router.delete( "/:id", orderController.cancelOrder );

module.exports = router;
```

1.2.4. productCategoryRoute.js

```
const express = require( "express" );

const upload = require( "../middleware/multer" );

const router = express.Router( );
```

```
const productCategoryController =
require("../controllers/productCategoryController");

router.post(  "/",upload.single("image"),
productCategoryController.createProductCategory);

router.get("/", productCategoryController.getAllCategories);

module.exports = router;
```

1.2.5. productRoute.js

```
const express = require("express");

const router = express.Router( );

const upload = require("../middleware/multer");

const productController = require("../controllers/productController");

router.post("/", upload.single("image"), productController.createProduct);

router.get("/", productController.getProducts);

router.get("/popular", productController.getPopularProducts);

module.exports = router;
```

1.2.6. promotionroute.js

```
const express = require("express");

const router = express.Router( );

const promotionController = require("../controllers/promotionController");

router.post("/", promotionController.createPromotionForProduct);
```

```
router.put("/:productId/apply", promotionController.applyPromotionToProduct);

router.put("/update-status", promotionController.updatePromotionStatus);

router.get("/promotion-products", promotionController.getPromotionProducts);

module.exports = router;
```

1.3. Define Model

1.3.1 User.js

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

const userSchema = new mongoose.Schema(
  {
    username: { type: String, required: true },
    email: { type: String, required: true },
    password: { type: String, required: true },
    fullname: { type: String, required: true },
    address: { type: String },
    phoneNumber: { type: String },
    image: { type: String },
  },
  { versionKey: false }
);
```

```
const User = mongoose.model( "User", userSchema );

module.exports = User;
```

1.3.2. Promotion.js

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

const promotionSchema = new mongoose.Schema( {

  productId: { type: mongoose.Schema.Types.ObjectId,ref: "Product",

    required: true },

  discountPercentage: { type: Number, required: true },

  startDate: { type: Date, required: true },

  endDate: { type: Date, required: true },

  isActive: { type: Boolean, default: true },

  createdAt: { type: Date, default: Date.now },} );

const Promotion = mongoose.model( "Promotion", promotionSchema );

module.exports = Promotion;
```

1.3.3. ProductCategory.js

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

const productCategorySchema = new mongoose.Schema(

  {
```

```
    cate_id: {
      type: mongoose.Schema.Types.ObjectId,
      default: mongoose.Types.ObjectId,
    },
    name: { type: String, required: true },
    description: { type: String },
    image: { type: String, required: true },
    createdAt: { type: Date, default: Date.now },
    updatedAt: { type: Date, default: Date.now },
  },
  { versionKey: false }
);
const ProductCategory = mongoose.model(
  "ProductCategory",
  productCategorySchema
);
module.exports = ProductCategory;
```

1.3.4. Product.js

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

```
const productSchema = new mongoose.Schema( {
  sku: { type: String, unique: true, required: true },
  name: { type: String, required: true },
  description: { type: String },
  price: { type: Number, required: true },
  cate_id: { type: mongoose.Schema.Types.ObjectId, required: true },
  quantity: { type: Number, required: true },
  image: { type: String, required: true },
  status: { type: String, enum: ["active", "inactive"], default: "active" },
  createdAt: { type: Date, default: Date.now },
  updatedAt: { type: Date, default: Date.now },
  discountedPrice: { type: Number, default: 0 },
  isOnPromotion: { type: Boolean, default: false },
  salesVolume: { type: Number, default: 0 },
} );
const Product = mongoose.model( "Product", productSchema );
module.exports = Product;
```

1.3.5 Order.js

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

const orderSchema = new mongoose.Schema({

userId: { type: mongoose.Schema.Types.ObjectId, ref: "User", required: true },

userName: { type: String, required: true },lat: { type: Number, required: true },

lng: { type: Number, required: true },phoneNumber: { type: String },

notes: { type: String },

status: {type: String,enum: ["pending", "processing", "completed"], default:"pending",},

  items: [

    {

      product: {

        type: mongoose.Schema.Types.ObjectId,

        ref: "Product",

        required: true,

      },

      quantity: { type: Number, required: true },

    },

  ],
```

```
  totalPrice: { type: Number, required: true },
  additionalInfo: { type: String },
  createdAt: { type: Date, default: Date.now },
});
const Order = mongoose.model("Order", orderSchema);
module.exports = Order;
```

1.3.6. Notification.js

```
const mongoose = require("mongoose");
const notificationSchema = new mongoose.Schema({
message: { type: String, required: true },
createdBy: {type: mongoose.Schema.Types.ObjectId,ref: "User",required: true},
read: { type: Boolean, default: false }, createdAt: { type: Date, default: Date.now },
});
const Notification = mongoose.model("Notification", notificationSchema);
module.exports = Notification;
```

1.4. Define Controller

1.4.1. authController

```
const User = require("../model/User");
const jwt = require("jsonwebtoken");
```

```
const bcrypt = require( "bcrypt" );

const authController = {};
```



```
phone: user.phoneNumber};

    res.status( 200 ).json( { status: "success", token, user: userResponse } );

  } catch ( error ) {

    console.error( "Error in login:", error );

    res.status( 500 ).json( { status: "error", message: "Internal server error" } );

  }

};

module.exports = authController;
```

1.4.2. notificationController.js

```
const Notification = require( "../model/Notification" );

exports.createNotification = async ( req, res ) => {

  try {

    const { message, createdBy } = req.body;
```

```
    const notification = new Notification( {
      message,
      createdBy, } );
    await notification.save( );
    res.status( 201 ).json( {
      message: "Notification created successfully",
      notification,
    } );
  } catch ( error ) {
    console.error( "Error creating notification:", error );
    res.status( 500 ).json( { message: "Internal server error" } );
  }
};
```



```
    res.status( 200 ).json( { message: "Notification read status updated", notification } );
  } catch ( error ) {
    console.error( "Error updating notification read status:", error );
```

```
    res.status( 500 ).json( { message: "Internal server error" } );
  }
```

1.4.3. OrderController.js

```
const axios = require( "axios" );
const Order = require( "../model/Order" );
const User = require( "../model/User" );
const Product = require( "../model/Product" );
async function sendMessage( chatId, text ) {
const botToken = "6823978357:AAHZn1uoJ6W__yqEghnsaPql7BFG7cveDgU";
const apiUrl = `https://api.telegram.org/bot${botToken}/sendMessage`;
```



```
exports.getTotalSales = async ( req, res ) => {
  try {
    const result = await Order.aggregate( [
      { $match: { status: "completed" } },
      { $group: { _id: null, totalSales: { $sum: "$totalPrice" } } },
```

```
    ]);

    const totalSales = result.length > 0 ? result[0].totalSales : 0;
    res.json({ totalSales });
  } catch (error) {
    console.error("Error calculating total sales:", error);
    res.status(500).json({ message: "Internal server error" });
  }
};
```

1.4.4. productCategoryController.js

```
const ProductCategory = require("../model/ProductCategory");
const mongoose = require("mongoose");
exports.createProductCategory = async (req, res) => {
  try {
    const { name, description } = req.body;
    const existingCategory = await ProductCategory.findOne({ name });
    if (existingCategory) {
      return res
        .status(400)
```

```
      .json( { message: "Category with the same name already exists" } );
  }
```



```
exports.getAllCategories = async ( req, res ) => {
 try {
  const categories = await ProductCategory.find( );
  res.status( 200 ).json( { categories } );
 } catch ( error ) {
  console.error( "Error fetching categories:", error );
  res.status( 500 ).json( { message: "Internal server error" } );
 }
};
```

1.4.5. productController.js

```
const Product = require( "../model/Product" );
const ProductCategory = require( "../model/ProductCategory" );
```

```
const Promotion = require("../model/Promotion");

const Order = require("../model/Order");

exports.createProduct = async (req, res) => {

  try {

    const { name, description, price, cate_id, quantity, status } = req.body;

    console.log("createProduct", req.body);

    const category = await ProductCategory.findById(cate_id);
```



```
    const products = await Product.find({

      status: "active",

      salesVolume: { $gte: minSalesVolume },

    })

      .sort({ salesVolume: -1 })

      .limit(limit);


    res.json({ message: "Popular products retrieved successfully", products });
```

```
  } catch (error) {
    console.error("Error retrieving popular products:", error);
    res.status(500).json({ message: "Internal server error" });
  }
```

1.4.6. promotionController.js

```
const Promotion = require("../model/Promotion");
const Product = require("../model/Product");
exports.createPromotionForProduct = async (req, res) => {
  try {
    const { productId, discountPercentage, startDate, endDate, isActive } =
      req.body;
    const start = new Date(`${startDate}T00:00:00Z`);
    const end = new Date(`${endDate}T00:00:00Z`);
    const existingPromotion = await Promotion.findOne({
      productId,isActive: true,
    });
```

```
    const promotionProducts = promotions.map( ( promo ) => ( {
      product: promo.productId,
      discountPercentage: promo.discountPercentage,
      discountedPrice:
        promo.productId.price * ( 1 - promo.discountPercentage / 100 ),
    } ) );
    res.status( 200 ).json( { promotionProducts } );
  } catch ( error ) {
    console.error( "Error fetching promotion products:", error );
    res.status( 500 ).json( { message: "Internal server error" } );
  }};
```

2- User Front-End

2.1 Home Screen

```
function Product( ) {
  const errRef = useRef<IErrDialogRef>( null );
  const navigate = useNavigate( );
  const { setCategories } = useAuthContext( );
  const [callData, setCallNewData] = useState( 1 );
  const { cart } = useContext( CartContext )!;
```

```
  const [produtDetail, setProdudctDetail] = useState<Iproduct.Product>( );
  const [drawerOpen, setDrawerOpen] = React.useState( false );
```



```
      <ProductDrawer
        drawerOpen={drawerOpen}
        setDrawerOpen={setDrawerOpen}
        productDetail={produtDetail}
      />
    </Box>
  );
}
export default Product;
```

2.2. Popular Product

```
function PopularProduct( ) {
  const navigate = useNavigate( );
  const errRef = useRef<IErrDialogRef>( null );
  const { cart, getTotalPrice, getProductIds } = useContext( CartContext ) !;
```

```
  const theme = useTheme( );

  const { t } = useTranslation( );

  const [produtDetail, setProdudctDetail] = useState<Iproduct.Product>( );

  const [drawerOpen, setDrawerOpen] = useState( false );
```



```
{cart.length > 0 && (
        <Box position={"fixed"} bottom={7} width={"100%"} px={2}>
          <CheckoutButton
            count={getProductIds( ).length || 0}
            total={getTotalPrice( )}
            handleOnClick={( ) => {
              navigate( ROUTE_PATH.cart );
            }}
          />
        </Box>
      )}
```

2.3. sign up

```
const SignupForm = ( ) => {

  const { control, handleSubmit } = useForm<IForm>( );

  const errRef = useRef<IErrDialogRef>( null );

  const navigate = useNavigate( );
```



```
  <Button

        type="submit"

        fullWidth

        variant="contained"

        sx={{

          textTransform: "uppercase",

          color: "background.paper",

          display: "flex",
```

```
            alignItems: "center",
            mt: 3,
          }}
        >
          Sign Up
        </Button></Box></Box> )}
export default SignupForm;
```

2.4. Login

```
const Login = ( ) => {const errRef = useRef<IErrDialogRef>( null );
  const { control, handleSubmit, watch } = useForm<Iform>( );
  const { setAuthState } = useAuthContext( );
  const { runAsync: runLogin, data: login,loading: loadingLogin,
  } = useRequest( Auth.login, {
    manual: true,onSuccess: ( data ) => {
      if ( data.token ) {
        setAuthState( {
          isLogIn: true,lan: "en", rememberMe: watch( "rememberMe" ),
```

```
        token: data?.token, name: data.user.username,userId: data.user.id,}); }
    } });
```



```
<Button
        type="submit" fullWidth variant="contained" disabled={loadingLogin}
         sx={{ textTransform: "uppercase",color: "background.paper",
           display: "flex", alignItems: "center",mb: 2, }} > {!loadingLogin ? ("Sign in"
         ) : (<Box sx={{ display: "flex", justifyContent: "center" }}>
            <LoadingSpiner size={25} /></Box>)} </Button>
       <Link to={`${ROUTE_PATH.singup}`}>Register new account</Link>
     </Box></Box>);
};
export default Login;
```

2.5. Search Product

```
function Search( ) { const errRef = useRef<IErrDialogRef>(null);
  const theme = useTheme( ); const navigate = useNavigate( );
```

```
const { categories } = useAuthContext( ) const { t } = useTranslation( );
const [searchProduct, setSearchProduct] = useState( "" );
const { loading: loadingList, data: allProduct } = useRequest(
  ( ) => PRODUCT_API.listProduct( undefined, searchProduct ),
  { onSuccess: ( data ) => {
      console.log( "SuccessRes", data );
    },ready: searchProduct !== "", refreshDeps: [searchProduct],
  }
);
```



```
<Grid container px={2} spacing={1}>
 {allProduct && allProduct.products.map( ( product, index ) => (
  <Grid item xs={6} key={product._id}>
  <ProductCard image={product.image} price={product.price} name={product.name}
  _id={product._id} description={product.description} cate_id={product.cate_id}
  quantity={product.quantity} status={product.status} createdAt={product.createdAt}
 updatedAt={product.updatedAt}__v={product.__v} width="auto"/> </Grid> ) )}
```

```
    </Grid>)}</>)} </Box></Box>);
}
export default Search;
```

2.5. Order

```
function Order( ) {
  const { authState } = useAuthContext( );
  const [orderId, setOrderId] = useState("");
  const openRef = useRef<ICusDialogHandler>(null);
  const {data: listUserOrdered, refresh: refreshOrder,loading: loadingLiseOrdered,
}=useRequest( ( )=>ORDER_API.getListUserOrder("669608f473a9fd5486460d1b")
, {
    refreshDeps: [authState?.userId],
  });
```



```
{loadingLiseOrdered ? (
      <Box sx={{ display: "flex", justifyContent: "center", alignItems: "center",
          height: "calc(100vh - 200px)", }}><LoadingSpiner size={25} /></Box>
```

```
      ) : listUserOrdered?.length !== 0 ? ( listUserOrdered &&listUserOrdered.map( ( i,
index ) => {return (

<Box key={i._id} sx={{ px: 2, mt: 2, mb: index ? 10 : 0 }}>

<OrderItem key={i._id} items={i.items} status={i.status} notes={i.notes} title="Online
Shop" totalPrice={i.totalPrice} createdAt={moment( i.createdAt ).format(

"MMMM Do YYYY, h:mm:ss a" )} orderId={i._id} setOrderId={setOrderId} /> </Box>

          ); } )

      ) : (


export default Order;
```

2.6. Order Detail

```
function OrderDetail( ) {

const openRef = useRef<ICusDialogHandler>( null );

const orderId = new URLSearchParams( window.location.search ).get( "orderId" );

const { isLoaded } = useJsApiLoader( {id: "google-map-script",googleMapsApiKey:
process.env.REACT_APP_MAP_KEY || "",libraries,} );

  const { data: orderDetail } = useRequest(

    ( ) => ORDER_API.orderDetail( orderId || "" ),

    {

      ready: orderId !== undefined,
```

```
    onSuccess: (data) => {
      console.log("dataOrderedDetail", data);
      openRef.current?.close( );
    },
  }
);
```



```
      <Marker position={ { lat: orderDetail?.lat || -1, lng: orderDetail?.lng || -1 } ||
          defaultCoord }/></GoogleMap></Box>
    <SwipeableEdgeDrawer orderDetail={orderDetail} /></Box>
  );
}
export default OrderDetail;
```

2.7. Checkout

```
function Checkout( ) {const theme = useTheme( ); return (<Box>
    <Toolbar sx={{ position: "sticky", top: 2, zIndex: 1, background: "white" }}>
```

```
        <IconButton edge="start" color="default"
          aria-label="back"
          onClick={( ) => {
            window.history.back( );
          }}
        >
```



```
  <ArrowBackIcon /> </IconButton>
        <Typography
          variant="h6"
          style={{ flexGrow: 1, textAlign: "center" }}
          color={theme.palette.grey["700"]}
        >
          Checkout
        </Typography>
      </Toolbar>
      <CheckoutContent /></Box> );}
export default Checkout;
```

2.8 Product Category

```
function CategoryList( ) {
  const errRef = useRef<IErrDialogRef>( null );
  const currentId = useParams( );
  const [produtDetail, setProdudctDetail] = useState<Iproduct.Product>( );
  const [drawerOpen, setDrawerOpen] = useState( false );
  const { cart, getTotalPrice, getProductIds } = useContext( CartContext )!;
  const navigate = useNavigate( );
  console.log( "id", currentId.id );
  const [cateId, setId] = useState( currentId.id || "" );
```



```
  <Box sx={{height: "calc( 100vh - 120px )", display: "flex",
            justifyContent: "center",
            alignItems: "center",
          }}
```

```
        >
          <Typography>No Data</Typography>
        </Box>
      )}
    </Box>
  );
}
export default CategoryList;
```

2.9. Cart

```
function Cart( ) { const { cart, getTotalPrice, getProductIds, clearCart } =
useContext( CartContext )!; const navigate = useNavigate( );
const openRef = useRef<ICusDialogHandler>( null ); return (
<Box sx={{ px: 2, overflow: "scroll", pb: 2 }}> <CusDialog maxWidth="lg"
ref={openRef}> <Box><Typography sx={{ mt: 2, ml: 2, mb: 2 }}>
Are you sure you want to clear your cart ? </Typography>
</Box><Box sx={{ display: "flex", justifyContent: "flex-end", pr: 2, pb: 2 }}>
<Button color="error" onClick={( ) => openRef.current ?.close( )}>No</Button>
<Button color="primary" variant="contained" onClick={( ) => { clearCart( );
 openRef.current ?.close( ); }} >Ok</Button></Box></CusDialog>
```

```
      <CheckoutButton
        count={getProductIds( ).length || 0}
        total={getTotalPrice( )}
        handleOnClick={( ) => {
          navigate( ROUTE_PATH.checkout );
        }}
      />
    </Box> );};
export default Cart;
```

2.10 Account

```
const Sidebar = ( ) => {
  const navigate = useNavigate( );
  const { t } = useTranslation( );
```

```
  const theme = useTheme( );

  const { authState, setLogInStatus } = useAuthContext( );

  const [open, setOpen] = useState<boolean>( false );

  const handleClickOpen = ( ) => { setOpen( true );};

  const handleClose = ( ) => {setOpen( false );};

  const { switchColorMode } = useContext( ThemeContext );

  const { changeLng } = useAuthContext( );const changeLanguage = ( lng: string ) =>
{changeLng( lng ); };
```



```
<Box textAlign="center" mb={5}><Typography variant="body1" style={{ fontWeight:
"bold" }}>{t( "FOLLOW US" )}</Typography><Box display="flex"
justifyContent="center" mt={1}> <IconButton href="#" style={{ color: "#3b5998" }}>

<FacebookIcon /></IconButton> <IconButton href="#" style={{ color: "#0088cc" }}>

<TelegramIcon /></IconButton></Box><Typography variant="body2"
color="textSecondary" mt={2}>Address: #S69 & S71, St 1031, Sangkat Khmuonh,
Sen Sok, Phnom Penh,CAMBODIA. </Typography></Box></Box> );

};

export default Sidebar;
```

2.11. Route

```
const router = createBrowserRouter( [{ path: ROUTE_PATH.login, element: (

<Suspense><Login /></Suspense> ),},{path: ROUTE_PATH.checkout,element: (

<Suspense><Checkout /></Suspense> ),},{path: ROUTE_PATH.about_us,

      element: ( <Suspense><Aboutus /></Suspense> )}{path: ROUTE_PATH.term,

element: ( <Suspense><Term /></Suspense> )}
```



```
{ path: ROUTE_PATH.account, element: ( <Suspense><Account /></Suspense> ),},

 {path: ROUTE_PATH.inbox,

          element: (

            <Suspense>

              <Inbox />

            </Suspense>

          ),},  ],

  { basename: process.env.REACT_APP_PUBLIC_URL } );

export { router }
```

3- **Admin Front-End**

3.1. Home

```
const DashboardItem = ( {title,count,icon,}: {title: string;count: string;icon ?: string;} )
=> { return (  <Paper elevation={0} sx={{ p: 2 }}> <Stack direction='row'
justifyContent='space-between'alignItems='center'><Stack><TypographygutterBottom
sx={{fontSize: ['body2.fontSize', 'body2.fontSize', 'body1.fontSize'],}}>{title}
</Typography>{icon  ?  ( <Avatar variant='square' src={icon} sx={{ width: [30, 30,
40], height: 'auto', aspectRatio: '1/1' }}/> ) : ( <Typography fontWeight='bold'
variant='h6'>{count} </Typography> )} </Stack>
```



```
<Grid container spacing={2}>{dataOrderActive ?.statusCounts.map( ( status, i ) => (
<Grid key={i} item xs={4}><DashboardItem
                {...{
                  title: status.status,
                  count: status.count.toString( ),
                }}
              /></Grid> ) )}</Grid></Paper></Grid></Grid>  );};
export default Home;
```

3.2. Account

```
const Account = ( ) => {
  const [succOpen, setOpenSucc] = useState( false );
  const [accOpen, setaccOpen] = useState( false );
  const method = useForm<IformAcc>( );
  const errRef = useRef<IErrDialogRef>( null );
  const onSubmit: SubmitHandler<IformAcc> = async ( data ) => {
    console.log( 'Data', data );
    const formData = new FormData( );
    formData.append( 'name', data.name );
    formData.append( 'email', data.email );
    formData.append( 'username', data.userName );
  };
```



```
<Box sx={{display: 'flex',flexDirection: 'row',justifyContent: 'flex-end',pt: 3,px: 3,}}>
<Button variant='contained' onClick={( )=>{
 method.handleSubmit( onSubmit )( ); }}>
```

```
Save</Button>

</Box>

    </Box>

  );

};

export default Account;
```

3.3 Login

```
const Login = ( ) => {

  const errRef = useRef<IErrDialogRef>( null );

  const { control, handleSubmit, watch } = useForm<IFormInputs>( );

  const { authState, setAuthState } = useAuthContext( );
```



```
 <Box display='flex' flexDirection='column'

alignItems='center' mt={1}>

 <Button disabled={loadingLogin}

variant='contained' type='submit'

size='large'
```

```
 fullWidth sx={{bgcolor: 'primary.light',
color: 'primary.main',
fontWeight: 'bold',}}>
 {!loadingLogin ? 'Login' :
<LoadingSpiner size={26.25} />}
</Button>
</Box>
        </form>
      </Paper>
    </Container>
  );};
export default Login;
```

3.4. Order

```
const Order = ( ) => {
  const [orderId, setId] = useState( '' );
  const [open, setOpen] = useState( false );
  const [pickData, setPickDate] = useState( false );
  const [activeTab, setActiveTab] = React.useState( 0 );
  const [active, setActive] = React.useState( 0 );
```

```
  const [history, setHistory] = useState(0);

  const errRef = useRef<IErrDialogRef>(null);

  const [activeOrder, setActiveOrder] = React.useState('activeOrder');

  const [startDate, setStartDate] = useState<string>('');

  const [endDate, setEndDate] = useState<string>('');

  const [mapCenter, setMapCenter] = useState<ICoord>(defaultCoord);

  const [address, setAddress] = useState<string>( );

  const searchRef = useRef<ISearchRef>(null);

  const [location, setLocation] = useState<string>('');

  const mediumDown = useMediaQuery(theme.breakpoints.down('md'));

  const currentDate = new Date( ).toISOString( ).split('T')[0];

  const [order, setOrder] = React.useState<string>('order');

  const orderStatus = activeOrderStatus[active].value;

  const ordreHistoryStatus = historyOrderStatus[active].value;
                                          ┆
                                          ┆

<TrackingPagedetailId={orderId} status={data?.status}

  date={data?.createdAt} orderTracking={[]} />) : null} </>} /</Box></Grid></Grid>

    </Paper);};

export default Order;
```

3.5. Product

```
const Product = ( ) => {
  const { urlParams, navigate } = useRouter( );
  const active =
    !urlParams.tab || urlParams.tab === ':tab'  ? 'product' : urlParams.tab;


  return (
    <Box sx={{ px: [2, 2, 3] }}>
      <StyledTabs
        value={active}
        onChange={( _, newVal ) =>
          navigate( ROUTE_PATH.product.replace( ':tab', newVal.toString( ) ), {
            replace: true,
          } )
        }
      >
        <StyledTab label='Product' value='product' />
      </StyledTabs>
      <TabPanel value={active} index='product'>
```

```
        <ProductTab />
      </TabPanel>
    </Box>
  );
};
export default Product;
```

3.6. Shop

```
function Shop( ) {const [active, setActive] = useState( 0 );return (
    <Box m={{ xs: 2, md: 3 }}><AppBar position='static' elevation={0}
        sx={{ background: ( theme ) => theme.palette.background.paper,
borderTopRightRadius: '10px',
 borderTopLeftRadius: '10px', mt: 3, }} >
        <>
          <StyledTabs value={active} onChange={( e, n ) => setActive( +n )}>
            <StyledTab  label='Shop Info'
sx={{ flex: 1, maxWidth: '50px', py: 1 }}
            /> <StyledTab label='Payment
sx={{ flex: 1, maxWidth: '50px', py: 1 }}
```

```
            />
          </StyledTabs>
        </>
      </AppBar>
  <TabPanel value={active} index={0}>
        <ShopInfo />
      </TabPanel>
      <TabPanel value={active} index={1}>
        <Payment />
      </TabPanel>
    </Box>
  );
}
export default Shop;
```

3.7. Route

```
const Login = lazy( ( ) => import( 'pages/Login' ) );
const Home = lazy( ( ) => import( 'pages/Home' ) );
const Order = lazy( ( ) => import( 'pages/Order' ) );
const Product = lazy( ( ) => import( 'pages/Product' ) );
```

```
const Shop = lazy( ( ) => import( 'pages/Shop' ) );

const Account = lazy( ( ) => import( 'pages/Account' ) );

const router = createBrowserRouter(

  [{path: ROUTE_PATH.login,element: (

        <Suspense>

          <Login />

        </Suspense>

      ),

    },

    {

      path: ROUTE_PATH.root,

      element: <Layout />,

      errorElement: <Navigate to={ROUTE_PATH.home} />,

      children: [

        {

          path: ROUTE_PATH.home,

          element: (

            <Suspense>

              <Home />
```

```
    </Suspense>
  ),
},
{
  path: ROUTE_PATH.order,
  element: (
    <Suspense>
      <Order />
    </Suspense>
  ),
},
{
  path: ROUTE_PATH.product,
  element: (
    <Suspense>
      <Product />
    </Suspense>
  ),
},
```

```
{
  path: ROUTE_PATH.shop,
  element: (
    <Suspense>
      <Shop />
    </Suspense>
  ),
},
{
  path: ROUTE_PATH.account,
  element: (
    <Suspense>
      <Account />
    </Suspense>
  ),
},
{
  path: ROUTE_PATH.root,
  element: <Navigate to={ROUTE_PATH.home} replace />,
```

```
        },
      ],
    },
  ],
  { basename: process.env.REACT_APP_PUBLIC_URL },
);
export { router };
```

3.8. Them

```
const primary: PaletteColorOptions = {
  main: '#016DEE',
  light: '#e4f0ff',
  dark: '#0157be',
};
const theme = createTheme( {
  palette: {
    primary,
    secondary: {
      main: '#22224B',
    },
```

```
info: {
  main: '#E0FF00',
},
warning: {
  main: '#FF6C0E',
},
success: {
  main: '#30B200',
  light: '#C0DF88',
},
error: {
  main: '#F5333F',
  light: '#FDE4E4',
},
background: {
  default: '#fff',
  paper: '#f5f5f5',
},
text: { primary: '#000000DE', secondary: '#616161' },
```

```
    action: {
      active: '#C9C9C8',
      selected: '#C9C9C8',
    },
    grey: { 50: '#eff0f5', 300: '#E3E3E2', 400: '#AFAFAF' },
  },
  shape: { borderRadius: 6 },
});
theme.shadows[2] = 'rgba(100, 100, 111, 0.2) 0px 7px 29px 0px';
theme.shadows[3] = '0px 4px 4px 0px #00000040';
theme.components = {
  MuiCssBaseline: {
    styleOverrides: ``,
  },
  MuiPaper: {
    styleOverrides: {
      rounded: {
        borderRadius: theme.spacing(2),
      },
```

```
  },
},
MuiMenu: {
  styleOverrides: {
    paper: {
      borderRadius: theme.spacing(0.5),
      backgroundColor: theme.palette.background.default,
    },
  },
},
MuiAppBar: {
  styleOverrides: {
    colorDefault: {
      backgroundColor: theme.palette.background.default,
    },
  },
},
MuiChip: {
  styleOverrides: {
```

```
    filled: {
      color: theme.palette.text.secondary,
      backgroundColor: theme.palette.grey[50],
    },
    filledPrimary: {
      color: theme.palette.common.white,
      backgroundColor: theme.palette.primary.main,
    },
  },
},
MuiButtonBase: {
  defaultProps: {
    disableRipple: true,
  },
},
MuiButton: {
  defaultProps: {
    disableElevation: true,
  },
```

```
    styleOverrides: {
      root: {
        textTransform: 'none',
        fontWeight: 700,
      },
      contained: {
        ':disabled': {
          background: theme.palette.action.active,
        },
      },
    },
  },
};
theme.typography.body1 = {
  fontSize: '0.875rem',
  [theme.breakpoints.up( 'md' )]: {
    fontSize: '1rem',
  },
};
```

```
theme.typography.body2 = { fontSize: '0.75rem', [theme.breakpoints.up( 'md' )]: {
    fontSize: '0.875rem',},};
theme.typography.caption = {fontSize: '0.625rem', [theme.breakpoints.up( 'md' )]: {
    fontSize: '0.75rem',},
};
export default theme;
```

Full Source Code and How to Run Project

1- **Full Source Code of Admin**
   - git clone https://github.com/BrandoWeaver/online-shop-admin.git
   - npm install
   - npm start
2- **Full Source Code of User**
   - git clone https://github.com/BrandoWeaver/online-shop-user.git
   - npm install
   - npm start
3- **Full Source Code of Backend**
   - git clone https://gitlab.com/srunrotha614/online-shop-api.git
   - npm install
   - npm run dev