សាកលវិទ្យាល័យភូមិន្ទនីតិសាស្ត្រ
និងវិទ្យាសាស្ត្រសេដ្ឋកិច្ច

សារណាបញ្ចប់ការសិក្សា

គេហទំព័រ ONLINE FORUM

ស្រាវជ្រាវ ពីថ្ងៃទី ១៦ ខែ មីនា ដល់ថ្ងៃទី ១៥ ខែ ឧសភា ឆ្នាំ២០២០

តាក់តែងឡើងដោយ
និស្សិតឈ្មោះ **ហួត ចាន់ស្រីនិច**

សាស្ត្រាចារ្យណែនាំ
បណ្ឌិត **ង៉ិន ហោច**

ថ្នាក់បរិញ្ញាបត្រ សេដ្ឋកិច្ចព័ត៌មានវិទ្យា
ជំនាន់ទី ១៥

ឆ្នាំចូលសិក្សា ២០១៦
ឆ្នាំសរសេរសារណា ២០២០

អារម្ភកថា

នៅក្នុងបរិបថសង្គមថ្មីបច្ចេកវិទ្យាបានដើរតួយ៉ាងសំខាន់ នៅក្នុងជីវភាពប្រចាំថ្ងៃរបស់មនុស្សយើងម្នាក់ៗ។ ជាងនេះនៅទៀតក្នុងដំណើរការនៃការសិក្សាស្រាវជ្រាវ សិស្សនិស្សិតគ្រប់រូបរមែងតែងជួបនូវបញ្ហានិងព្យាយាមស្វែងរកដំណោះស្រាយដ៏សមស្របមួយ។ ហេតុនេះនាងខ្ញុំដែលជានិស្សិតដែលសិក្សានៅ សាកលវិទ្យាល័យភូមិន្ទនីតិសាស្ត្រ និងវិទ្យាសាស្ត្រសេដ្ឋកិច្ច ជំនាញ សេដ្ឋកិច្ចពត៌មានវិទ្យា បានធ្វើការសិក្សាស្រាវជ្រាវទៅលើប្រធានបទមួយ " គេហទំព័រ Online Forum" ។

ការសិក្សាស្រាវជ្រាវនេះគឺជាគម្រោងមួយដែល នាងខ្ញុំបានស្រាវជ្រាវ ដើម្បីជួយសម្រួលដល់ ប្អូនៗ សិស្សានុសិស្សដែល រៀនមុខជំនាញ និងមានចំណាប់អារម្មណ៏ទាក់ទងនឹងមុខវិជ្ជានានា ក្នុងការបញ្ចេញមតិនិងចោទជាសំនួរផ្សេងៗក្នុងការសិក្សារបស់ពួកគេ។

បន្ទាប់ពីបញ្ចប់នូវការស្រាវជ្រាវរួចរាល់ហើយនាងខ្ញុំ បានទទួលចំណេះដឹងយ៉ាងច្រើន ក្នុងដំនើរការនៃការបង្កើតជារបសាយដែល អនុញ្ញាតិអោយអ្នកប្រើប្រាស់អាចបញ្ចេញមតិនិងចែករំលែកចំណេះដឹងបាន។

ជាចុងបញ្ចប់នឹងខ្ញុំ សង្ឃឹមថាការសិក្សាស្រាវជ្រាវ លើប្រធានបទមួយនេះពិតជាអាចជួយដល់ការអនុវត្តមួយចំនួនផងដែរ។

យើងខ្ញុំជឿជាក់ថា ការចងក្រងនូវសៀវភៅមួយនេះឡើង រមែងមានចំណុចខ្វះខាតជាច្រើនទៀតដែលនាងខ្ញុំផ្ទាល់ពុំបានគិតដល់ ដូច្នេះហើយនាងខ្ញុំសូមទទួលស្វាគមន៍ជានិច្ចនូវរាល់មតិយោបល់ និងការរិះគន់នានាក្នុងន័យស្ថាបនា ពីសំណាក់ លោកសាស្ត្រាចារ្យ លោកគ្រូ អ្នកគ្រូ និងមិត្តសិស្ស និស្សិតទាំងអស់ដើម្បីកែប្រែឲ្យកាន់តែប្រសើរឡើង។

នាងខ្ញុំសូមសំណូមពរដល់ សិស្សនិស្សិតដែលសិក្សាជំនាន់ក្រោយ សូមព្យាយាមសិក្សាស្រាវជ្រាវបន្ថែមរាល់ចំណុចខ្វះខាត នៅក្នុងផ្នែកណាមួយ និងខិតខំបង្កើតនូវគំនិតច្នៃប្រឌិតអោយបានច្រើនដើម្បីឲ្យប្រធានបទមួយនេះកាន់តែមានប្រសិទ្ធិភាពដើម្បីបានជាប្រយោជន៍សម្រាប់យើងទាំងអស់គ្នា។

សេចក្តីថ្លែងអំណរគុណ

នាងខ្ញុំ ហួត ចាន់ស្រីនិច ជានិស្សិតជំនាន់ទី ១៥ នៃសាកលវិទ្យាល័យភូមិន្ទនីតិសាស្ត្រ និងវិទ្យាសាស្ត្រសេដ្ឋកិច្ច ផ្នែកសេដ្ឋកិច្ចពត៌មានវិទ្យាឆ្នាំ សិក្សា ២០១៦-២០២០ នាងខ្ញុំសូមគោរពនិង ថ្លែងអំណរគុណយ៉ាងជ្រាលជ្រៅទៅដល់៖

លោកឪពុក អ្នកម្តាយ ដែលបានផ្តល់កំណើត និងខិតខំចិញ្ចឹមបីបាច់រក្សាកូន និងបានទំនុកបម្រុងកូនតាំងពីកើតរហូតដល់មានថ្ងៃនេះ។ ជាងនេះទៅទៀតលោកបានខិតខំប្រឹងប្រែង អោយកូនបានរៀនសូត្រ និងមានចំណេះដឹង ដែលអាចរស់នៅនិងបម្រើសង្គមជាតិមួយនេះបានប្រកបដោយភាពថ្លៃថ្នូរ។

លោកឯកឧត្តមសាកលវិទ្យាធិការ សាកលវិទ្យាធិការរង លោក និងលោកស្រីសាស្ត្រាចារ្យព្រមទាំងបុគ្គលិកនៃសាកលវិទ្យាល័យភូមិន្ទនីតិសាស្ត្រ និងវិទ្យាសាស្ត្រសេដ្ឋកិច្ចទាំងអស់ដែលបានខិតខំប្រឹងប្រែងយ៉ាងសកម្មនៅក្នុងបេសកកម្មបណ្តុះបណ្តាលធនធានមនុស្សគ្រប់ជំនាន់ ជាពិសេសលោក លោកស្រីបានអនុញ្ញាតអោយនាងខ្ញុំបានសិក្សា ក្រេបយកចំណេះដឹងរហូតដល់បញ្ចប់ថ្នាក់បរិញ្ញាបត្រមួយនេះ ។

លោកសាស្ត្រាចារ្យ បណ្ឌិត ទិន ហេង ដែលលោកជាសាស្ត្រាចារ្យដែលបានបង្ហាត់បង្រៀននាងខ្ញុំ ក៏ដូចជាបាន បង្ហាញនាងខ្ញុំ និងផ្តល់នូវឯកសារល្អៗ ជាច្រើនសម្រាប់ជាទុនក្នុងការស្រាវជ្រាវមួយនេះ។ លោកបានចំណាយពេល និងបានចំណាយទាំងកម្លាំងកាយនិងចិត្ត មិនថាស្ថិតក្នុងកាលៈទេសៈណាក៏ដោយ លោកតែងតែផ្តល់អនុសាសន៍ល្អៗដល់នាងខ្ញុំ។

បងប្អូន មិត្តរួមថ្នាក់ និងមិត្តរួមជំនាន់ទាំងអស់ដែលបានជ្រោមជ្រែង ផ្តល់កម្លាំងចិត្ត ដែលធ្វើអោយនាងខ្ញុំទទួលបានជោគជ័យ និងសរសេរសារណាមួយនេះបញ្ចប់ជាស្តាពរ។

ជាចុងក្រោយ នាងខ្ញុំសូមគោរពជូនពរ ចំពោះលោកឪពុក អ្នកម្តាយ លោកឯកឧត្តមសាកលវិទ្យាធិការ សាកលវិទ្យាធិការរង លោកសាស្ត្រាចារ្យបណ្ឌិត ទិន ហេង ក៏ដូចជាលោក លោកស្រីសាស្ត្រាចារ្យនៃ សាកលវិទ្យាល័យភូមិន្ទនីតិស្ត្រ និងវិទ្យាសាស្ត្រសេដ្ឋកិច្ចទាំងអស់ រួមទាំង បងប្អូន និងមិត្តរួមជំនាន់ទាំងអស់ សូមអោយជួបប្រទះតែសេចក្តីសុខ និងពុទ្ធពរទាំងបួនប្រការ គឺ អាយុ វណ្ណៈ សុខៈ ពលៈ កុំបីឃ្លៀងឃ្លាតឡើយ។

មាតិកា

សេចក្តីផ្តើម

ជំពូកទី១៖ រំលឹកទ្រឹស្តីដែលពាក់ព័ន្ធ

ជំពូកទី២៖ សម្រង់អត្ថបទដែលពាក់ព័ន្ធ

សេចក្តីសន្និដ្ឋាន

បញ្ជីតារាង

បញ្ជីរូប

សេចក្តីផ្តើម

១. ចំនាំបញ្ហានៃការស្រាវជ្រាវ

ស្ថិតក្នុងសតវត្សទីម្ភៃមួយ បច្ចកវិទ្យាហាក់បានដើរតួយ៉ាងសំខាន់ក្នុងការអភិវឌ្ឍន៍សង្គមជាតិមួយអោយមានភាពជឿនលឿននិងកាន់តែីកចម្រើនពីមួយថ្ងៃទៅមួយថ្ងៃ។ ជាងនេះទៅទៀតយើងសង្កេតឃើញថាបច្ចេកវិទ្យាបានដើរតួយ៉ាងសកម្មក្នុងវិស័យនានា ក្នុងប្រទេសកម្ពុជា រួមមាន វិស័យកសិកម្ម វិស័យសុខាភិបាល វិស័យអប់រំ និងវិស័យសំខាន់ៗផ្សេងៗទៀតផងដែរ។ ជាក់ស្តែងចំពោះវិស័យអប់រំ យើងសង្កេតឃើញថា ក្រសួងអប់រំបានខិតខំប្រឹងប្រែង បង្កើតនូវកម្មវិធីសិក្សានានាជាប្រព័ន្ធ Online ដើម្បីបង្កើននូវប្រសិទ្ធិភាពនៃការអប់រំ និង អនុញ្ញាតអោយសិស្សងាយស្រួលក្នុងការទទួលយកចំណេះដឹងថ្មីៗ។

ដោយឡែកកាលពីពេលកន្លងទៅថ្មីៗនេះ ដោយសារបញ្ហាកូវីដ-១៩ សាលារៀននៅទូទាំងប្រទេសបានបិទទ្វារជាហេតុដែលញុំាងអោយសាលារៀនមួយចំនួនធំបានសម្រេចចិត្តបង្រៀនអនឡាញ។ ទន្ទឹមនឹងនេះដែរនៅពេលសិក្សាតាមអនឡាញ សិស្សនិស្សិតតែងតែជួបប្រទះនូវបញ្ហាឬ ចម្ងល់ផ្សេងៗ ហេតុនេះសិស្សានុសិស្សត្រូវខិតខំប្រឹងប្រែងស្រាវជ្រាវនៅតាមអ៊ីនធឺណេត ឬសួរលោកគ្រូ អ្នកគ្រូទៅតាម បណ្តាញសង្គមនានា។ នៅពេលដែលមានសំណួរច្រើននៅក្នុងមុខវិជ្ជាតែមួយឬច្រើនសាស្ត្រាចារ្យ ត្រូវការចំណាយពេលច្រើន ក្នុងការឆ្លើយសំណួរ។ ការធ្វើបែបនេះ ហាក់បីដូចជាខាតពេលវេលា និងការចែកចាយនូវចំណេះដឹងហាក់ពុំសូវទូលំទូលាយនោះឡើយ។

ដោយយល់ឃើញបែបនេះហើយទើប ញុំាងអោយនាងខ្ញុំសម្រេចចិត្តបង្កើតនូវគេហទំព័រមួយដោយប្រមូលផ្តុំនូវសំណួរ និងចម្លើយនានាទៅតាម មុខវិជ្ជាផ្សេងៗដើម្បីជួយសម្រួលដល់ការសិក្សាស្រាវជ្រាវ ក៏ដូចជាបង្កើនការចែករំលែកចំណេះដឹងឲ្យកាន់តែទូលំទូលាយផងដែរ។

២ .ចំណោទបញ្ហានៃការស្រាវជ្រាវ

យើងសង្កេតឃើញថាការប្រើប្រាស់នូវ Forum បានជួយសម្រួលដល់អ្នកប្រើប្រាស់ យ៉ាងច្រើនក្នុងការចែករំលែក ចំណេះដឹង ជាងនេះទៅទៀត រាល់សំណួរនិងចម្លើយ ត្រូវបានប្រមូលផ្តុំទៅតាមមុខវិជ្ជា ជាហេតុនាំឲ្យអ្នកប្រើប្រាស់ងាយស្រួលក្នុងការស្វែងរកដំណោះស្រាយបានលឿន រហ័ស។

៣ .គោលបំណងនៃការស្រាវជ្រាវ

គោលបំណងនៃការស្រាវនេះធ្វើឡើងដើម្បីជួយពន្លឿនការសិក្សា របស់សិស្ស និស្សិតអោយកាន់តែមានប្រសិទ្ធិភាព។ តាមរយៈការប្រើប្រាស់ Online Forum លោកគ្រូ អ្នកគ្រូអាចធ្វើការឆ្លើយសំណួរជាលក្ខណៈទូទៅបាន។ ការធ្វើបែបនេះអាចអោយសិស្សនិស្សិត ទទួលបានចំណេះដឹងសម្បូរបែបពីការប្រើប្រាស់ forum។

៤. ដែនកំណត់និងវិសាលភាពនៃការស្រាវជ្រាវ

ដោយសារពេលវេលានៃការស្រាវជ្រាវមានកំណត់ហេតុនេះស្ថិតក្នុងដំណើរការនៃការបង្កើតគេហទំព័រមួយនេះឡើងនាងខ្ញុំបានបង្កើតនូវមុខងារចម្បងនិងសំខាន់ៗមួយចំនួនរួមមាន៖

- អនុញ្ញាតអោយអ្នកប្រើប្រាស់ធ្វើការចុះឈ្មោះតាមរយៈអ៊ីមែល
- អនុញ្ញាតអោយអ្នកប្រើប្រាស់ស្វែងរកនូវអត្ថបទនីមួយៗទៅតាមមុខវិជ្ជា
- អ្នកប្រើប្រាស់អាចបញ្ចេញមតិទៅលើអត្ថបទនីមួយៗបាន
- អ្នកប្រើប្រាស់អាចបង្កើតមុខវិជ្ជាថ្មីដែលមិនទាន់មាន
- អនុញ្ញាតអោយអ្នកប្រើប្រាស់ សរសេរនូវអត្ថបទថ្មីៗ
- អ្នកប្រើបាស់អាចបោះឆ្នោតទៅលើអត្ថបទណាមួយដែលពួកគេ ចូលចិត្ត
- ផ្តល់នូវផ្ទាំងសរសេរអត្ថបទដែលជួយសម្រួលដល់អ្នកប្រើប្រាស់ក្នុងការសរសេរអត្ថបទទៅតាមទម្រង់ដែលគាត់ចង់បាន។

៥. សារៈសំខាន់នៃការស្រាវជ្រាវ

តាមរយៈការប្រើប្រាស់គេហទំព័រមួយនេះ បានផ្តល់នូវគុណសម្បត្តិដល់អ្នកប្រើប្រាស់ ដូចជា៖

- អនុញ្ញាតអោយមានការចែករំលែកចំណេះដឹងកាន់តែទូលំទូលាយ
- ងាយស្រួលក្នុងការស្វែងរកដំណោះស្រាយនានា
- អ្នកប្រើប្រាស់អាចសួរជាសំណួរផ្សេងៗដើម្បីអោយអ្នកប្រើប្រាស់ផ្សេងៗទៀតផ្តល់នូវយោបល់
- បង្កើននូវទំនាក់ទំនងនៃអ្នកប្រើប្រាស់នីមួយៗ
- អ្នកប្រើប្រាស់អាចសួរ និងឆ្លើយសំណួរនានា បានគ្រប់ពេលនិងគ្រប់ទីកន្លែង។

ជំពូកទី១
រំលឹកទ្រឹស្តីដែលពាក់ព័ន្ធ

១ .១.អ្វីជា Framework?

Framework ឬ Software Framework សម្តៅទៅលើ platform ដែលគេអាចធ្វើការអភិវឌ្ឍន៍កម្មវិធីណាមួយបាន។ មិនតែប៉ុណ្ណោះវាបានផ្តល់មូលដ្ឋាន និងមុខងារស្រាប់ មួយចំនួនដែលជួយសម្រួលអ្នកអភិវឌ្ឍន៍កម្មវិធីចំណាយពេលតិចក្នុងការសរសេរកូដឡើងវិញ។ ការប្រើប្រាស់ Framework ជួយអោយអ្នកប្រើប្រាស់ចំណេញពេលច្រើនក្នុងការបង្កើតកម្មវិធីជាក់លាក់ណាមួយ។

១ .២.អ្វីជា Frontend Framework?

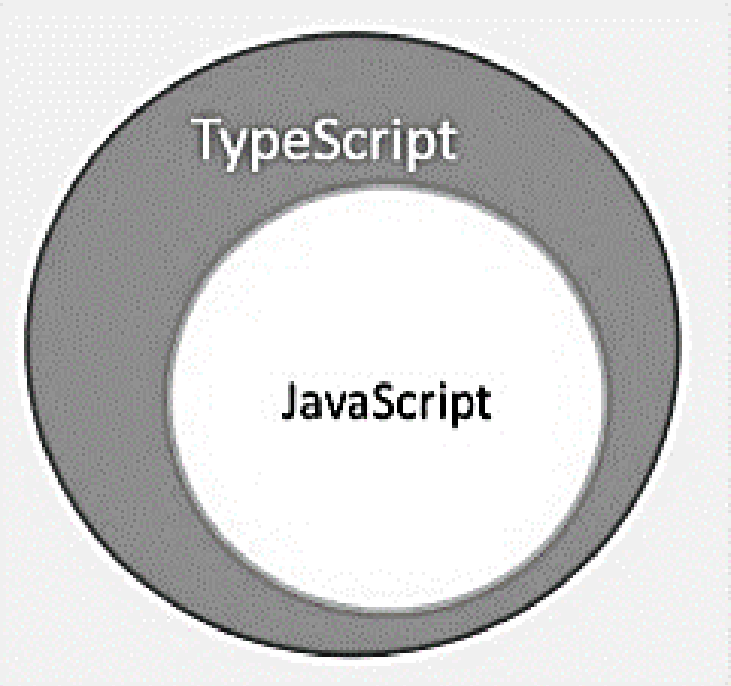
Frontend Framework ជាការប្រើប្រាស់នូវមូលដ្ឋានផ្នែក client side ពោលគឺការប្រើប្រាស់នូវ Hypertext Markup Language (HTML) , Cascade Style Sheet (CSS), Javascript រួមបញ្ចូលគ្នា បង្កើតបានជាផ្នែកតូចៗនៃវេបសាយ។ អ្នកប្រើប្រាស់អាចធ្វើការហៅកូដដែលមានស្រាប់ យកមកប្រើប្រាស់តាមតម្រូវការរបស់ខ្លួន។ សព្វថ្ងៃនេះគេសង្កេតឃើញថា Frontend Framework មួយចំនួនដែលគេពេញនិយមប្រើរួមមាន៖

React.js, Angular, Vue.js, jQuery, Backbone.js ។ល។

១ .៣.អ្វីជាភាសាប្រូក្រាម TypeScript?

TypeScript ជាភាសាប្រូក្រាមមួយដែលត្រូវបានអភិវឌ្ឍ ដោយលោក **Anders Hejlsberg** និងដាក់ឲ្យប្រើប្រាស់ដោយ ក្រុមហ៊ុន Microsoft ។ វាសង្កត់ធ្ងន់ទៅលើ syntax របស់វាស្រដៀងនឹងភាសាប្រូក្រាមមួយចំនួនផងដែរ។ ជាងនេះទៅទៀតលក្ខណៈ Syntax របស់ស្រដៀងទៅនឹង ភាសា JavaScript រួមនឹងភាសា Java Programming ។ គេអាចហៅម្យ៉ាងទៀតថា TypeScript មានផ្ទុកនូវ JavaScript ហេតុនេះហើយ កម្មវិធីមួយចំនួនដែលប្រើប្រាស់នូវ JavaScript គេអាចយកវ៉ាមកប្រើប្រាស់ជាមួយនឹង TypeScript បាន។ ជារួម TypeScript ជាភាសាមួយដែល៖

- មានប្រភេទទិន្នន័យច្បាស់លាស់
- ប្រើប្រាស់Object ជាគោល (Object Oriented)
- ជាភាសាប្រូក្រាមមួយដែលធ្វើការ compile មុននឹងដំណើរការ។



រូបភាពទី១ ការបង្ហាញពី TypeScript និង JavaScript

មូលហេតុមួយចំនួនដែលគេគួរប្រើនូវភាសាមួយនេះព្រោះ៖

- Compilation៖ ជារួមមុនពេលនូវកម្មវិធីដំណើរការវាធ្វើការសិក្សាទៅលើកូដម្តងមួយបន្ទាត់ៗ ក្នុងករណីណាដែលវារកឃើញថាមាននូវបញ្ហាណាមួយកើតឡើងនោះ វានឹងធ្វើការបញ្ជាក់ប្រាប់នូវបញ្ហានោះយ៉ាងជាក់លាក់មុននឹងវាដំណើរការ។
- Strong Static Typing៖ បាននិយាយថា វាមាននូវប្រភេទទិន្នន័យរបស់វាជាក់លាក់ដែលអាចអោយគេងាយស្រួលក្នុងការប្រើប្រាស់និងគណនាផ្សេងៗ។

១ .៤. អ្វីជា Angular Framework?

Angular គឺជា Frontend Framework មួយដែលប្រើប្រាស់សម្រាប់Single Page Application ដែលធាតុផ្សំនៃ Framework មួយនេះគឺបង្កើតឡើងដោយប្រើប្រាស់នូវ Hypertext Markup Language (HTML) និង TypeScript។ Angular ត្រូវបានប្រើប្រាស់នូវ ធាតុសំខាន់ៗនៃភាសា TypeScript មកដំណើរការ Frameworkទាំងមូល។ គ្រឹះសំខាន់នៃដំណើរការ Framework មួយនេះត្រូវបានបែងចែកជាផ្នែកតូចៗដែលគេហៅថាModule (NgModule)។ ជារួម NgModule ដើរតួជាអ្នកគ្រប់គ្រងនូវ componentនៃ Angular និងអនុញ្ញាតអោយមានការ compile នៃ application។ នៅក្នុងAngularតែងតែមានយ៉ាងហោចណាស់មួយនៃ

module ដើម ដែលmoduleដើមមួយនេះអនុញ្ញាតិអោយកម្មវិធីទាំងមូលដំណើរការបាន។ ការប្រើប្រាស់នូវ Angular Framework បានផ្តល់នូវផលប្រយោជន៍មួយចំនួនដូចជា៖

- ផ្តល់នូវភាពងាយស្រួលក្នុងការរៀបចំនូវគ្រោងកម្មវិធីទាំងមូល
- ងាយស្រួលក្នុងការកែប្រែនៅពេលក្រោយ
- សរសេរកូដតិច កាត់បន្ថយការសរសេរកូដដដែលៗ
- ងាយស្រួលក្នុងការតេស្តរកនូវបញ្ហានានា មុននឹងដាក់អោយប្រើប្រាស់។

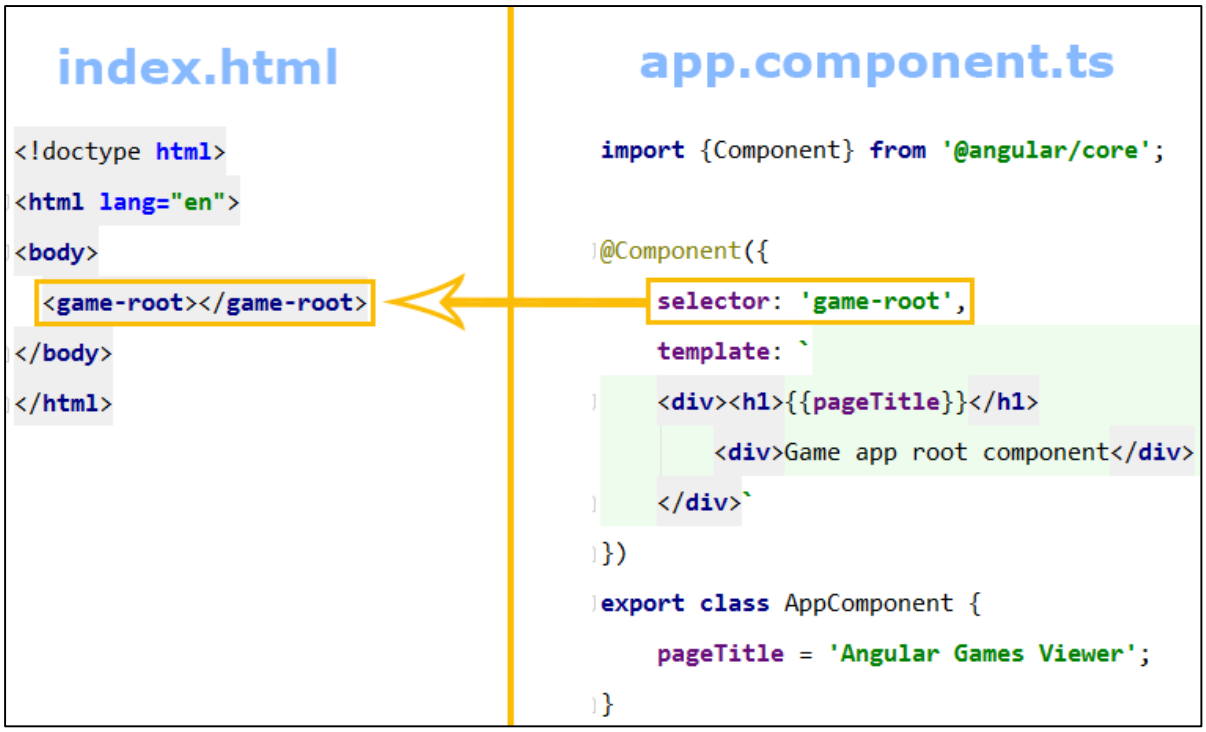
រូបភាពទី២ រូបតំណាងនៃ Angular Framework

១.៤.១. Angular Component

Component ជាផ្នែកសំខាន់មួយនៅក្នុងកម្មវិធី Angular។ វាផ្ទុកនូវ និយមន័យនៃ View ណាមួយ និង ទិន្នន័យនៃView រួមទាំងលក្ខណៈនៃ view ផងដែរ។ Angular Component មានទម្រង់ដូចជា Class មួយនៃ

JavaScriptដែលបានកំណត់ដោយ ប្រើប្រាស់ decorator @Component។ និមិត្តសញ្ញានេះផ្តល់អោយ component នូវ View សម្រាប់បង្ហាញមកកាន់ អ្នកប្រើប្រាស់និងពត៌មានមួយចំនួនទាក់ទងនឹង component ។ ដោយឡែក View សម្តៅលើ អ្វីដែលបង្ហាញទៅកាន់អ្នកប្រើប្រាស់តាមរយៈ HTML។ គេអាចហៅ component ថាជាធាតុតូចៗនៃ កម្មវិធីទាំងមូល។ component មានធាតុបីសំខាន់ៗរួមមាន៖

- Template: កំណត់នូវរូបរាងនៃ component ដើម្បីបង្ហាញទៅកាន់អ្នកប្រើប្រាស់។
- Class:ផ្នែកដែលផ្ទុកនូវអនុគមន៍ សម្រាប់ដំណើរការនៃ component វាត្រូវបានបង្កើតតាមរយៈ TypeScript Class។
- និង Metadata: ផ្តល់នូវពត៌មានលម្អិតអំពី component។



រូបភាពទី៣ ការប្រើប្រាស់ Component

១.៤.២. Angular Service

Service ជាបណ្តុំកូដ ដែលគេអាចហៅយកទៅប្រើប្រាស់តាមគោលបំណងជាក់លាក់ណាមួយ។ ជាទូទៅ service មួយត្រូវបានហៅយកទៅប្រើប្រាស់ ពីច្រើន component។ គេប្រើប្រាស់ service ក្នុងគោលបំណង៖

- ចែករំលែក នូវលក្ខណៈនិងទិន្នន័យមួយចំនួនទៅកាន់ component នានា
- ដើរតួជាអ្នកគ្រប់គ្រងទិន្នន័យពីផ្នែកខាងក្រៅនៃកម្មវិធី

- អាចយកទៅប្រើប្រាស់នូវក្នុង moduleផ្សេងៗបាន
- ងាយស្រួលហៅទៅប្រើប្រាស់
- សរសេរកូដតិច
- មានភាពងាយស្រួលក្នុងការសិក្សាលក្ខណានានា ជាជាងការសិក្សាលក្ខខណ្ឌដោយផ្ទាល់នៅក្នុង Component។

Service ជាclassមួយដែលមានextension .service.ts ។ រាល់ដំណើរការនានាជាមួយនឹង ការប្រើប្រាស់គេប្រើប្រាស់Service។ មុននឹងអាចប្រើប្រាស់នូវserviceបានគេត្រូវធ្វើការប្រកាសវានៅក្នុង constructor មួយនៅក្នុង class component ជាមុនសិន។ មិនតែប៉ុណ្ណោះស្ទើគ្រប់ប្រព័ន្ធដែលប្រើប្រាស់ នូវ Angular Framework គេតែងកត់សម្គាល់ឃើញថា វាត្រូវបានប្រើប្រាស់ក្នុងការទាញទិន្នន័យពីខាង Server Side។ រាល់ទិន្នន័យនានាដែលទទួលបានពី Service និងត្រូវបានយកទៅរៀបចំតាមបែបផែនផ្សេងៗទៅតាមរូបរាងនៃ គេហទំព័រ។ ជាទូទៅ Service ធ្វើការហៅទិន្នន័យបានតាមរយៈការប្រើប្រាស់នូវ HttpClient ដែលជាservice នៅក្នុង module HttpClientModule។ គេអាចបង្កើត នូវ service បានដោយប្រើប្រាស់ នូវពាក្យបញ្ជា មួយគឺ ng generate service (service name)។ លក្ខណៈសម្គាល់មួយទៀតនៃserviceគឺនៅលើ class service តែងមានភ្ជាប់មកជាមួយនូវ @Injectable។

```
import { Component } from '@angular/core';

import { ProductService } from './product.service';
import { Product } from './product';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
})

export class AppComponent
{

   products:Product[];
   productService;

   constructor(){
     this.productService=new ProductService();
   }

   getProducts() {

     this.products=this.productService.getProducts();
   }

}
```

រូបភាពទី៤ ការប្រើប្រាស់ service ក្នុង component

```
import {Product} from './Product'

export class ProductService{

    public  getProducts() {

        let products:Product[];

        products=[
            new Product(1,'Memory Card',500),
            new Product(1,'Pen Drive',750),
            new Product(1,'Power Bank',100)
        ]

        return products;
    }
}
```

រូបភាពទី៥ class Service

១.៤.៣. Angular Routing

Routing គឺជាការផ្លាស់ប្តូរពី ផ្នែកមួយទៅផ្នែកណាមួយផ្សេងទៀតនៃ Angular។ នៅក្នុងការប្រើប្រាស់គេត្រូវធ្វើការហៅវាពី @angular/router ។

```
const appRoutes={ path: 'product', component: ProductComponent }
```

រូបភាពទី៦ ការកំណត់ componentជាមួយroute

១.៥. អ្វីជា Spring Framework?

Spring គឺជា Open source application framework របស់ភាសា Java ។ វាមានផ្តុំកនូវ feature សំខាន់ៗនានាដែលគេនិយមប្រើប្រាស់នៅក្នុងភាសាJava។ វាត្រូវបានគេស្គាល់ថា ជា framework មួយដ៏ល្អសម្រាប់ប្រើប្រាស់ជាផ្នែកBackend នៃវេបសាយ។ Spring ផ្តល់នូវលក្ខណៈពិសេសជាច្រើនដូចជា៖

- ធ្វើការលឿនតាមរយៈការបម្លែងជា Bytecode
- ផ្តល់នូវ សុវត្ថិភាពខ្ពស់

- ផ្តល់នូវ លក្ខខណ្ឌសម្បូរបែបសម្រាប់វាយតម្លៃ
- ផ្តល់នូវ មុខងារមួយចំនួន ដែលផ្តល់ការងាយស្រួលក្នុងការសរសេរកូដ ដូចជា ទំនាក់ទំនងទៅកាន់ database ជាដើម។
- មានលក្ខណៈងាយស្រួលក្នុងការតេស្តនានា
- ធ្វើការលឿនដោយមិនប្រកាន់នូវ Operating System។

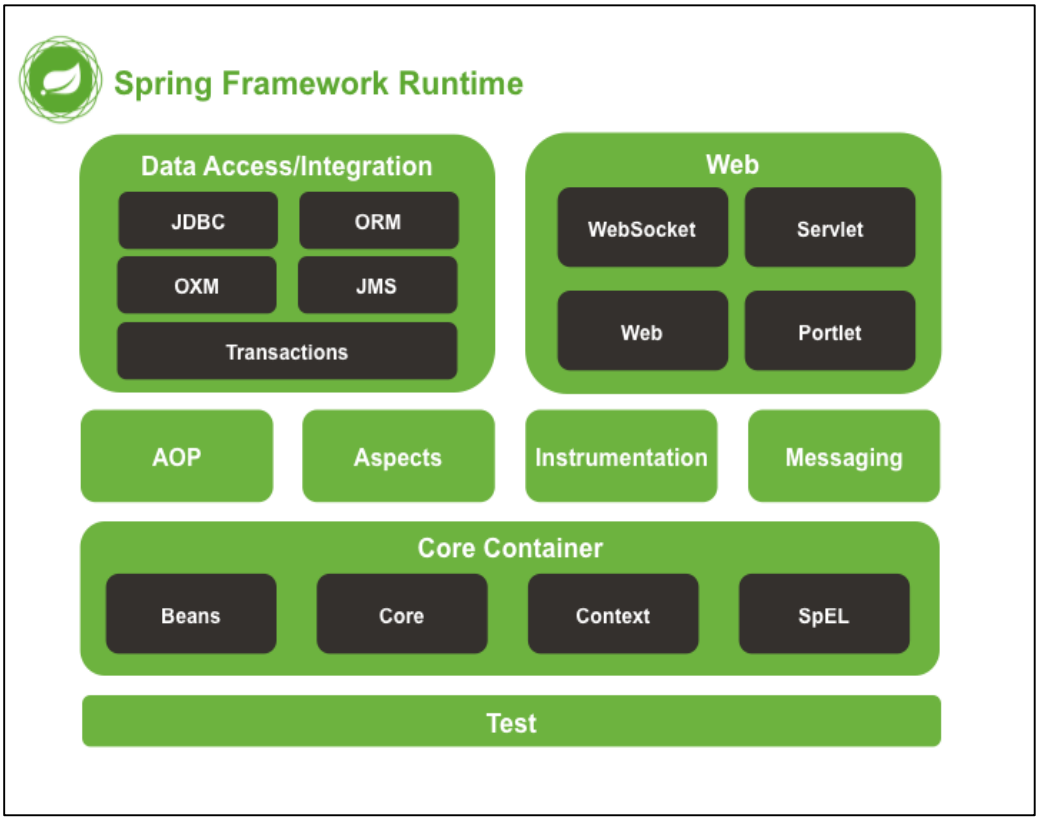
រូបភាពទី៧ រូបតំណាង Spring Framework

នៅក្នុង Spring Framework Module សំខាន់ៗរួមមាន៖

- Test៖ ផ្តល់នូវមុខងារមួយចំនួនសម្រាប់អោយគេពិនិត្យនូវកម្មវិធីមុននឹងដាក់ដំណើរការ តាមរយៈ JUnit និង TestNG។
- Spring Core Container៖ ផ្តល់នូវការគ្រប់គ្រងរាល់Objectទាំងឡាយតាំងពីការបង្កើត ការរក្សាទុក និងការលុបចោលទៅវិញ នៅក្នុង module នេះក៏មានផ្នែកតូចៗដូចជា៖
    - Core and Bean៖ Moduleនេះ ផ្តល់នូវការទីតាំងនៃការប្រើប្រាស់BeanឬObject ទាំងឡាយណានៅក្នុងកម្មវិធីទាំងមូល។
    - Context៖ ជាអ្នកផ្តល់នូវមុខងារពិសេសៗមួយចំនួនដូចជា ការបកប្រែភាសា ជាដើម។

- Expression Language៖ អនុញ្ញាតអោយអ្នកប្រើប្រាស់ធ្វើការសិក្សាទៅលើលក្ខខណ្ឌនានា។

- AOP, Aspects and Instrumentation៖ ផ្តល់នូវ ការប្រើប្រាស់ Advice, PointCut ជាដើម។
- Data Access / Integration៖ ដើរតួនាទីក្នុងការទាញយកនិងប្រើប្រាស់ទិន្នន័យពីប្រភពណាមួយ។
- Web៖ ផ្តល់នូវការបង្កើតនូវគេហទំព័រទាំងមូល។

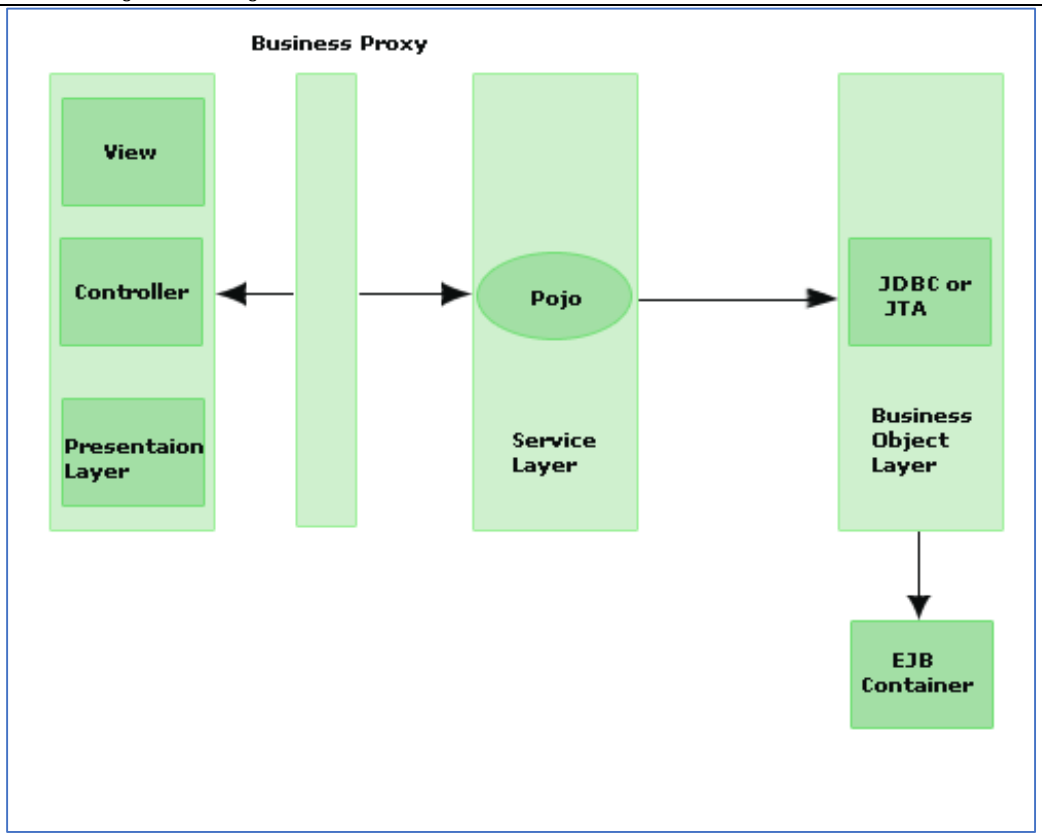


រូបភាពទី៨ Module សំខាន់ៗរបស់Spring Framework

១.៥.១. Spring POJO Programming Model

ជាការបែងចែកនូវទម្រង់នៃ Class ស្ថិតនៅក្នុងSpring ទៅតាមមុខងាររៀងៗខ្លួនរបស់វាដូចជា៖

- Data Access Layer៖ ជាអ្នកធ្វើការផ្ទាល់ជាមួយប្រភពទិន្នន័យ។
- Business Layer៖ សិក្សារាល់លក្ខខណ្ឌនានានៅក្នុងកម្មវិធី។
- Controller Layer៖ ធ្វើការផ្លូវផ្តងរាល់ការស្នើសុំនានាជាមួយអ្នកប្រើប្រាស់។
- Persistence/UI Layer៖ សម្របសម្រួលផ្ទាល់ជាមួយអ្នកប្រើប្រាស់។



រូបភាពទី៩ គម្រូនៃPOJO Programming Model

១.៥.២. Spring Data JPA

ជាផ្នែកមួយនៃ Spring Data JPA ផ្តល់នូវលក្ខណៈងាយស្រួលក្នុងការប្រើប្រាស់ និងការធ្វើការជាមួយទិន្នន័យពីក្នុងDatabase។ វាត្រូវបានប្រើប្រាស់ក្នុងការផ្គូផ្គងទិន្នន័យពីក្នុង Databaseជាមួយ class entity នៅក្នុង Java។ ដើម្បីអាចប្រើប្រាស់នូវ Spring Data JPAបានគេត្រូវធ្វើការភ្ជាប់ទៅកាន់ Databaseជាមួយសិនតាមរយៈការកំណត់នូវ DataSource និងការបន្ថែមនូវ dependencyរបស់វាទៅកាន់ pom.xml file។ ខាងក្រោមនេះជាគម្រូ នៃ Dependency នៃ Spring Data JPA៖

```
<dependency>
<groupId>org.springframework.data</groupId>
<artifactId>spring-data-jpa</artifactId>
<version>${spring.data}</version>
</dependency>
<dependency>
```

<groupId>org.hibernate</groupId>

<artifactId>hibernate-entitymanager</artifactId>

<version>${hibernate.manager}</version>

</dependency

```
BookRepository.java

package com.mkyong;

import org.springframework.data.repository.CrudRepository;

import java.util.List;

public interface BookRepository extends CrudRepository<Book, Long> {

    List<Book> findByName(String name);

}
```

រូបភាពទី១0 ការបង្ហាញពីការប្រើប្រាស់ Spring Data JPA

១.៥.៣. Spring Project Lombok

Lombok គឺជា Java Library ដែលប្រើប្រាស់ដើម្បីកាត់បន្ថយការសរសេរកូដនៅក្នុង class តំរូណាមួយ តាមរយៈការប្រើប្រាស់ annotation។ ដើម្បីអាចប្រើប្រាស់វាបានគេត្រូវ ដាក់ dependency វាចូលទៅក្នុង file មួយដែលមានឈ្មោះថា pom.xml ជាមុនសិន។ ជាងនេះទៅទៀតតាមរយៈការប្រើប្រាស់វា យើងមិនចាំបាច់បង្កើតនូវ getter setter និង constructor របស់ class នោះទេ។

```
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
</dependency>
```

រូបភាពទី១១ Dependency Lombok

```java
import lombok.AllArgsConstructor;
import lombok.NoArgsConstructor;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;

@NoArgsConstructor
@RequiredArgsConstructor
@AllArgsConstructor
public class Product {
    @NonNull private Integer id;
    private String name;
}
```

រូបភាពទី១២ ការប្រើប្រាស់Lombok

១.៦. អ្វីជា Postgresql?

Postgresql គឺជាប្រព័ន្ធគ្រប់គ្រង Database ដែលពឹងផ្អែកលើភាសាStructure Query Language (SQL)។ ទិន្នន័យនៅក្នុង Postgresql ត្រូវបានរក្សាទុកក្នុងតារាងដែលមានជួរដេក និងជួរឈរ។

រូបភាពទី១៣ បង្ហាញពីLogo Postgresql

ប្រភេទទិន្នន័យក្នុង Postgresql

Name	Aliases	Description
bigint	int8	signed eight-byte integer
bigserial	serial8	autoincrementing eight-byte integer
bit [(n)]		fixed-length bit string

Name	Aliases	Description
`bit varying [ (n) ]`	`varbit [ (n) ]`	variable-length bit string
`boolean`	`bool`	logical Boolean (true/false)
`box`		rectangular box on a plane
`bytea`		binary data ("byte array")
`character [ (n) ]`	`char [ (n) ]`	fixed-length character string
`character varying [ (n) ]`	`varchar [ (n) ]`	variable-length character string
`cidr`		IPv4 or IPv6 network address
`circle`		circle on a plane
`date`		calendar date (year, month, day)
`double precision`	`float8`	double precision floating-point number (8 bytes)
`inet`		IPv4 or IPv6 host address
`integer`	`int`, `int4`	signed four-byte integer
`interval [ fields ] [ (p) ]`		time span
`json`		textual JSON data
`jsonb`		binary JSON data, decomposed
`line`		infinite line on a plane
`lseg`		line segment on a plane
`macaddr`		MAC (Media Access Control) address
`money`		currency amount
`numeric [ (p, s) ]`	`decimal [ (p, s) ]`	exact numeric of selectable precision
`path`		geometric path on a plane
`pg_lsn`		PostgreSQL Log Sequence Number
`point`		geometric point on a plane
`polygon`		closed geometric path on a plane
`real`	`float4`	single precision floating-point number (4 bytes)
`smallint`	`int2`	signed two-byte integer

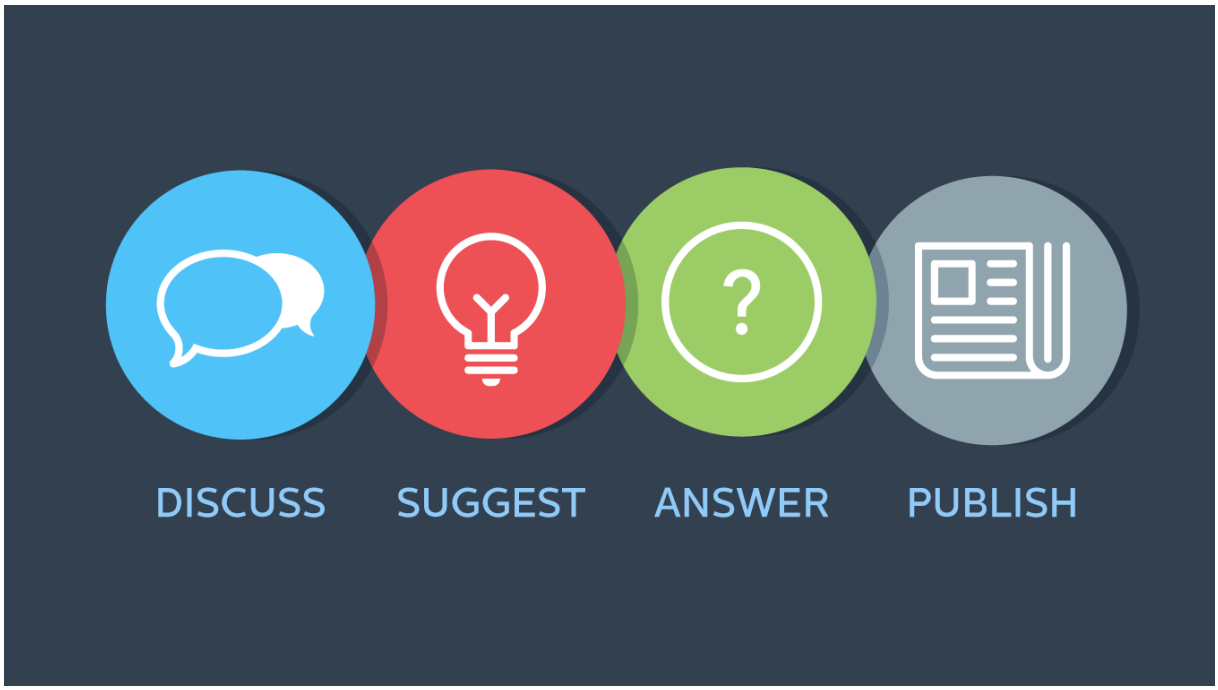
Name	Aliases	Description
smallserial	serial2	autoincrementing two-byte integer
serial	serial4	autoincrementing four-byte integer
text		variable-length character string
time [(p)] [without time zone]		time of day (no time zone)
time [(p)] with time zone	timetz	time of day, including time zone
timestamp [(p)] [without time zone]		date and time (no time zone)
timestamp [(p)] with time zone	timestamptz	date and time, including time zone
tsquery		text search query
tsvector		text search document
txid_snapshot		user-level transaction ID snapshot
uuid		universally unique identifier
xml		XML data

ជំពូកទី២

សម្រង់អត្ថបទដែលពាក់ព័ន្ធ

២ .១.ការណែនាំអំពី Online Forum

Online Forum ឬ Internet Forum គឺជា ការពិភាក្សាគ្នានិងផ្លាស់ប្តូរមតិគ្នាតាមរយៈអ៊ីនធឺណេត ទៅតាមការគិតឃើញ ទៅតាមប្រភេទអត្ថបទ និងប្រធានបទផ្សេងៗគ្នា។ ពួកគេក៏អាចសួរជាសំណួរ និងបំផុសនូវប្រធានបទបានផងដែរ។ ជាទូទៅនៅក្នុងការប្រើប្រាស់ Forum អ្នកប្រើប្រាស់ត្រូវបានបែងចែកជាពីរប្រភេទរួមមាន អ្នកប្រើប្រាស់ដែលបានចុះឈ្មោះហើយនិងអ្នកប្រើប្រាស់ដែលមិនទាន់បានចុះឈ្មោះ។ ភាគច្រើនចំពោះអ្នកប្រើប្រាស់ដែលពុះទាន់បានចុះឈ្មោះគឺគាត់មានសិទ្ធិបានត្រឹមតែអានតែប៉ុណ្ណោះ។



រូបភាពទី១៤ មុខងារនៃOnline Forum

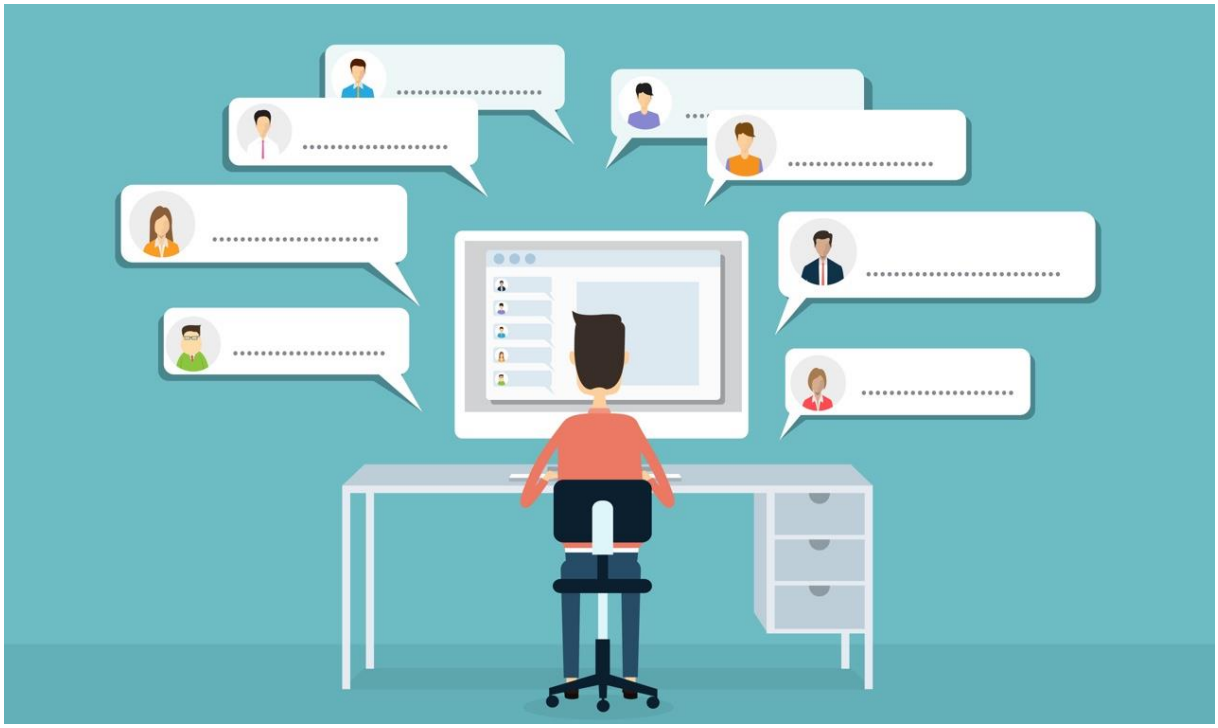
២ .២.ប្រភេទ្ធិនៃ Online Forum

នៅក្នុងយុគសម័យនៃការប្រើប្រាស់នូវ កុំព្យូទ័របានធ្វើអោយពិភពលោកកាន់តែតូចជាងមុនបានន័យថាការប្រាស្រ័យទាក់ទងគ្នា ពីមួយថ្ងៃទៅមួយថ្ងៃកាន់តែងាយស្រួល និងឆាប់រហ័ស។ សព្វថ្ងៃនេះពាក្យថាសហគមន៍នៅក្នុងប្រព័ន្ធអ៊ីនធឺណេតហាក់បីដូចជាធំជាងភូមិសាស្ត្រពិតប្រាកដ។ ជាងនេះទៅទៀតOnline Forum ត្រូវបានគេប្រៀបប្រដូចទៅនឹង សហគមន៍មួយនៃសម័យបច្ចេកវិទ្យា។

នៅប្រហែលមុនពេលការបង្កើតនូវ កុំព្យូទ័រផ្ទាល់ខ្លួនការចែករំលែកចំណេះដឹងគ្នាហាក់ពុំសូវមានលក្ខណៈជាសកលទូលំទូលាយឡើយ។គេប្រើប្រាស់នូវ "Bulletin Board" ដែលជាគេហទំព័រដែលអាចអោយអ្នកប្រើប្រាស់ឆ្លើយនូវសំណួរផ្សេងៗទៅតាមប្រធានបទដែលគេបានដាក់ អោយ។

តាមរយៈការប្រើប្រាស់នូវគេហទំព័រមួយនេះគេរកឃើញថាចំនួនអ្នកប្រើប្រាស់ដែលចូលចិត្តចូលរួមបញ្ចេញមតិនិងចែករំលែកចំណេះដឹងហាក់មានច្រើន គួរអោយកត់សម្គាល់ពីមួយថ្ងៃទៅមួយថ្ងៃ។ រហូតមកដល់សតវត្សទី២ម្តងមួយ ការប្រើប្រាស់មាតិកានេះនៅលើគេហទំព័រមានប្រជាប្រិយភាពយ៉ាងខ្លាំងដែលជាសញ្ញាណដ៏ល្អមួយ។

Online Forum បានកែប្រែជីវភាពរស់នៅរបស់មនុស្សយ៉ាងច្រើនពិសេសចំពោះ អាជីវករពួកគាត់ប្រើប្រាស់វាសម្រាប់ជាកន្លែងផ្តល់ប្រឹក្សាអាជីវកម្មនិងសេវាកម្ម សិស្សនិស្សិតអាចផ្លាស់ប្តូរមតិយោបល់និងចំណេះដឹងជាមួយគ្នា។ ហេតុនេះពីមួយថ្ងៃទៅមួយថ្ងៃមនុស្សបានចាប់ផ្តើមប្រើវាជាមធ្យោបាយមួយក្នុងការស្វែងរកដំណោះស្រាយ ឬស្វែងរកការសម្រេចចិត្តដ៏ត្រឹមត្រូវណាមួយ។



រូបភាពទី១៥ រូបតំណាងនៃការប្រើប្រាស់ Online Forum

២ .៣.ប្រភេទនៃ Online Forum

២. ៣.១ News Forum

ជា ប្រភេទForum ដែលគេប្រើប្រាស់ដើម្បីចែកចាយព័ត៌មានណាមួយទៅកាន់ក្រុមមនុស្សណាមួយ។ ជាទូទៅនៅក្នុងគេហទំព័រប្រភេទនេះអ្នកដែលចែកចាយអត្ថបទគឺមានមួយឬពីរអ្នកតែប៉ុណ្ណោះ។

២. ៣.២. Standard Forum

ជា ប្រភេទForum ដែលគេនិយមប្រើប្រាស់ គេហទំព័រប្រភេទនេះ អ្នកប្រើប្រាស់ទាំងអស់អាចសួរសំណួរ និងបញ្ចេញមតិបានដូចៗគ្នា ។

២. ៣.៣. Single Simple Forum

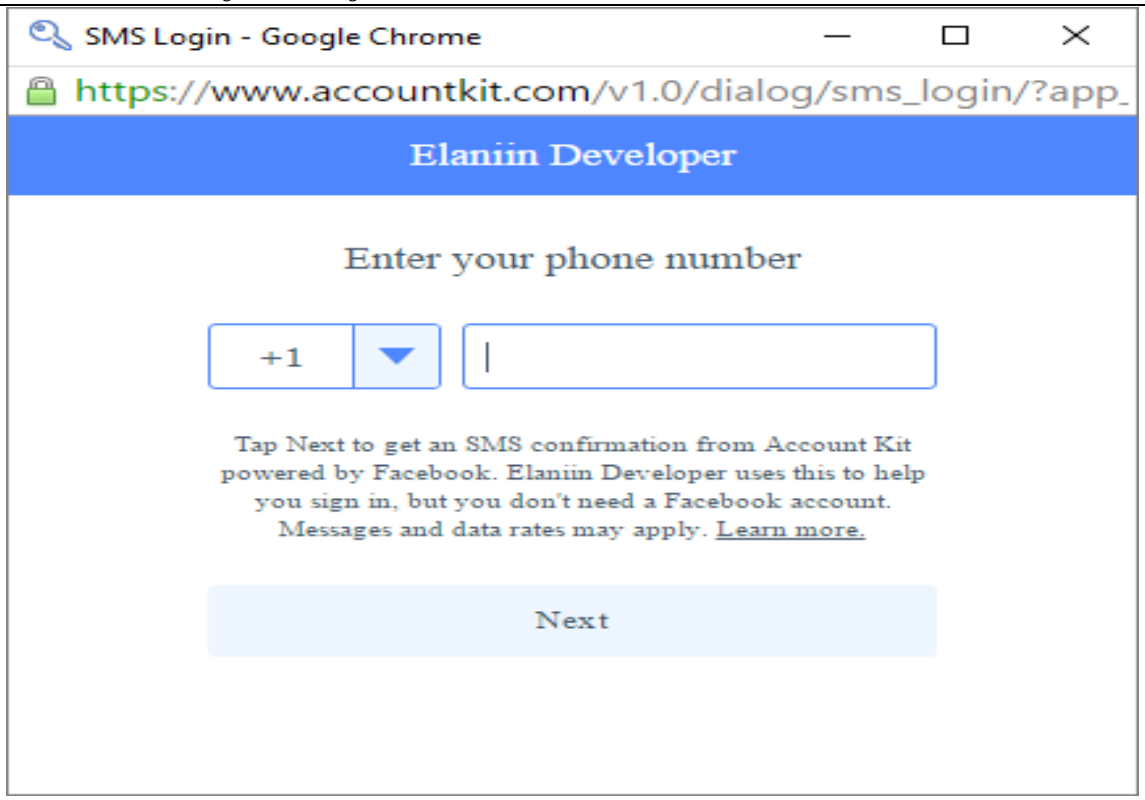
ជា ប្រភេទForum ស្រដៀងនឹង News Forumដែរ តែក្រៅពីអ្នកសួរសំណួរ អ្នកប្រើប្រាស់ដទៃទៀតចាំបាច់ត្រូវតែចូលឆ្លើយសំណួរ។

២.៤ .ដំណើរការនៃ Online Forum

២.៤.១. ការបង្កើតគណនីអ្នកប្រើប្រាស់

ដើម្បីងាយស្រួលក្នុងការគ្រប់គ្រងទិន្នន័យអ្នកប្រើប្រាស់ ចាំបាច់ណាស់ដែលអ្នកប្រើប្រាស់ត្រូវការបង្កើតគណនីថ្មីតាមរយៈការប្រើប្រាស់ អ៊ីមែល លេខទូរស័ព្ទជាដើម។ ដោយសារបច្ចេកវិទ្យារីកចម្រើនគេអាចជ្រើសរើសវិធីសាស្ត្រផ្សេងៗគ្នា ក្នុងចំណោមវិធីសាស្ត្រខាងក្រោមដើម្បីទាញយកព័ត៌មានមួយចំនួនពីអ្នកប្រើប្រាស់របស់ញាគេ វិធីទាំងនោះមាន៖

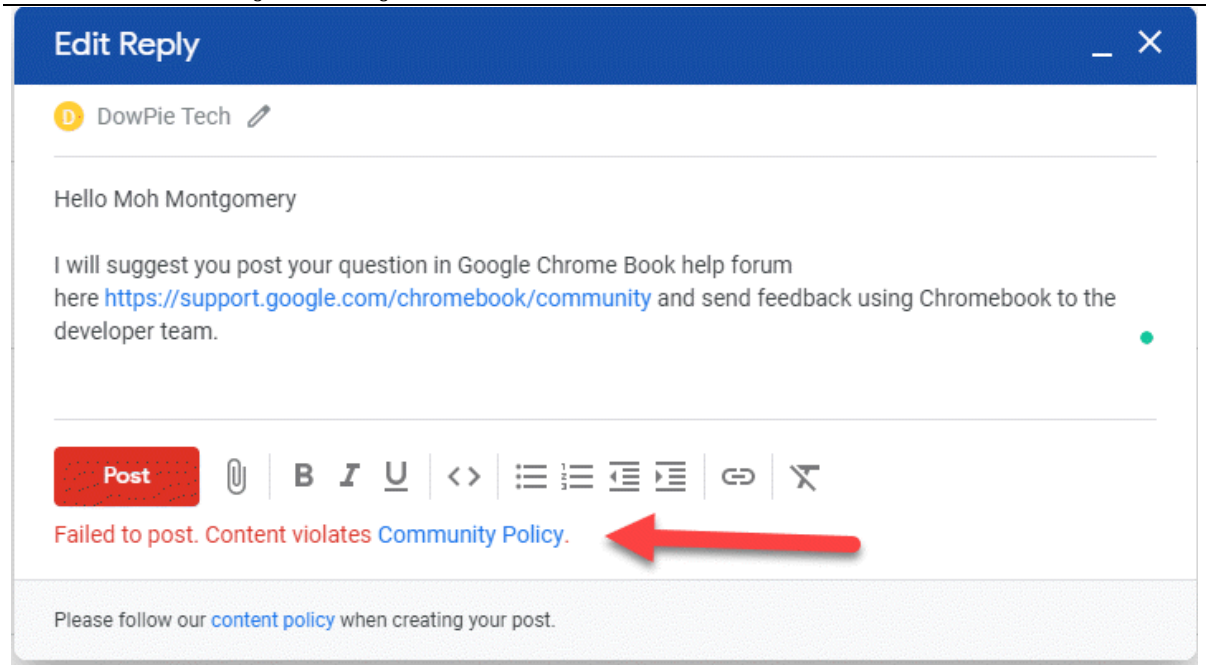
- ការផ្ទៀងផ្ទាត់លេខទូរស័ព្ទជាមួយ Firebase
- ការផ្ទៀងផ្ទាត់លេខទូរស័ព្ទ ឬ លេខកូដសម្ងាត់ដោយប្រើប្រាស់ Account Kit
- ការប្រើប្រាស់នូវ អ៊ីមែល ដើម្បីបញ្ជាក់អត្តសញ្ញាណអ្នកប្រើប្រាស់
- ការរក្សាទុកនូវអ៊ីមែលអ្នកប្រើប្រាស់ដោយផ្ទាល់ និងលេខសម្ងាត់
- ការជ្រើសរើសបណ្តាញសង្គមដើម្បីទាញយកទិន្នន័យអ្នកប្រើប្រាស់។



រូបភាពទី១៦ រូបតំណាងការប្រើប្រាស់លេខទូរស័ព្ទដើម្បីចុះឈ្មោះ

២.៤.២. ការបង្កើតទ្វារសំណួរ

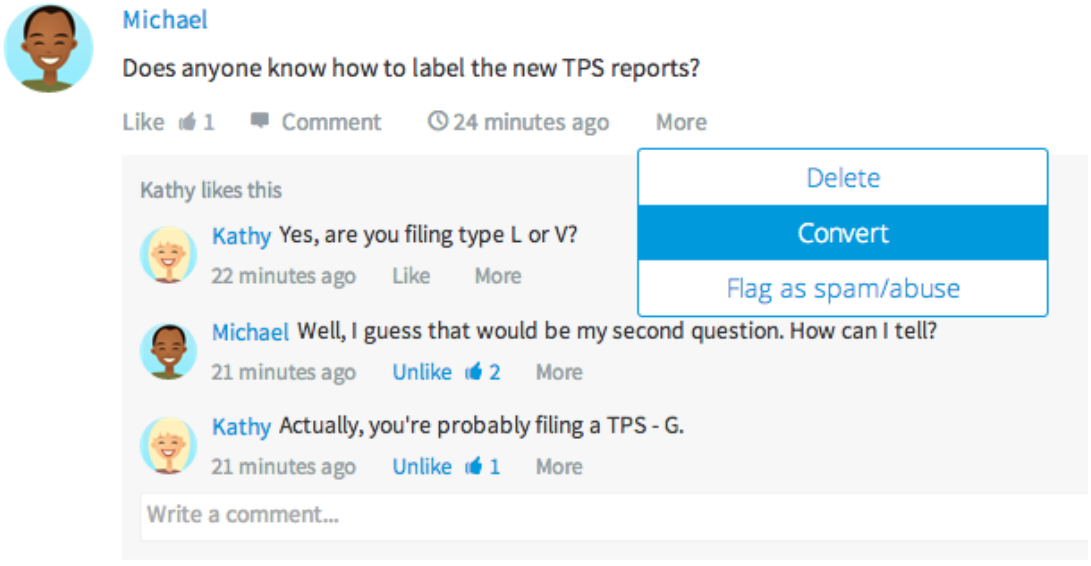
បន្ទាប់ពីមានគណនីគ្រប់គ្រាន់អ្នកប្រើប្រាស់អាចធ្វើការសួរសំណួរ ណាមួយ ដើម្បីអោយអ្នកប្រើប្រាស់ដទៃទៀតអាចចូលប្រើប្រាស់បាន។ យើងសង្កេតឃើញថាការប្រើប្រាស់ Online Forum អ្នកប្រើប្រាស់មានគោលបំណងធំមួយគឺការអាចជជែក ផ្តែកញែក ការសាកសុំមតិយោបល់បាន ដែលមុខងារនេះគេហៅថា ការបង្កើតសំណួរ។ ជារួមអ្នកប្រើប្រាស់ធ្វើការ បង្ហោះនូវសំណួររួចរង់ចាំការឆ្លើយតបនានាពីអ្នកប្រើប្រាស់ដ៏ទៃទៀតនៅក្នុងគេហទំព័រនេះផងដែរ។



រូបភាពទី១៧ រូបតំណាងការបង្ហោះអត្ថបទថ្មីណាមួយ

២.៤.៣. ការបញ្ចេញមតិទៅកាន់ប្រធានបទនានា

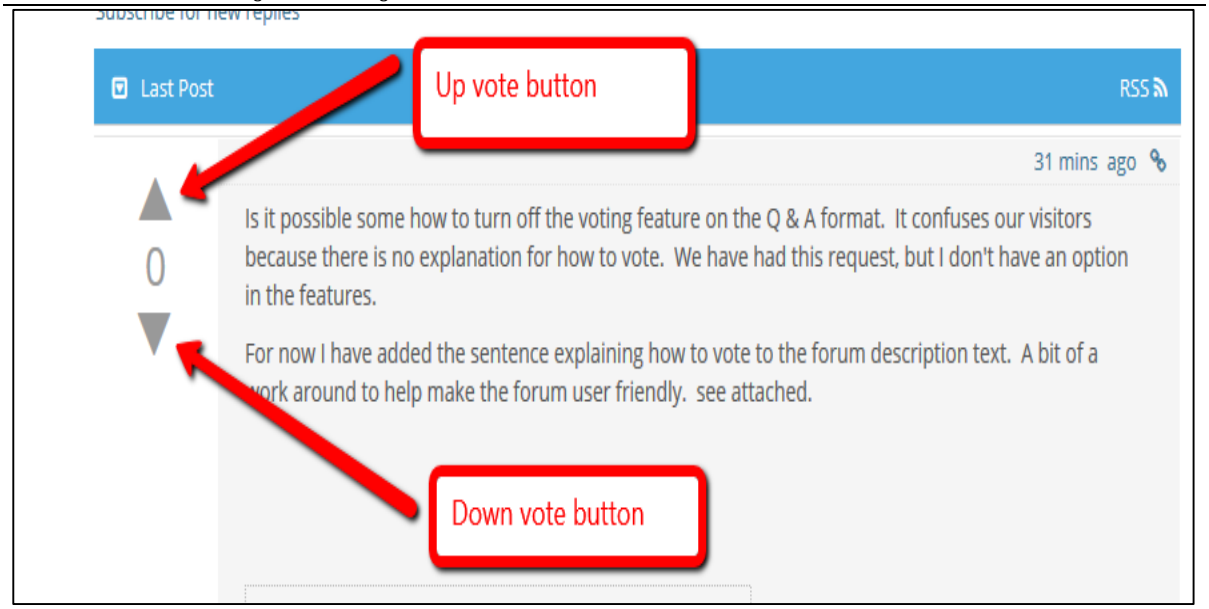
អ្នកប្រើប្រាស់អាចបញ្ចេញមតិយោបល់របស់គាត់ទៅកាន់សំណួរឬអត្ថបទណាមួយដែលគាត់ចង់បញ្ចេញយោបល់។



រូបភាពទី១៨ រូបតំណាងការបញ្ចេញមតិ

២.៤.៤. ការចោះឆ្លោតប្រធានបទ

ជាមុខងារដែលអ្នកប្រើប្រាស់ បង្ហាញពីការពេញចិត្ត ឬមិនពេញចិត្តចំពោះអត្ថបទណាមួយ។



រូបភាពទី១៩ រូបតំណាងការបោះឆ្នោត

ជំពូកទី៣

ជំណើរការនៃការស្រាវ

៣.១. ការវិភាគលើរចនាសម្ព័ន្ធវេបសាយ

ការជ្រើសរើសប្រធានបទគឺធ្វើឡើងតាមភាពចាំបាច់ជាក់ស្តែង ជាពិសេសស្ថិតក្នុងសម័យកាល ជំងឺកូវីដ១៩ ការស្រាវជ្រាវតាមអ៊ីនធឺណេតបានដើរតួជាកត្តចម្បងសម្រាប់សិស្សនិស្សិតក្នុងការសិក្សាស្រាវជ្រាវបន្ថែម ក៏ដូចជាស្វ័យសិក្សាផងដែរ។

៣ .២.ការវិភាគបម្រើបម្រាស់គេហទំព័រ

ដើម្បីបង្ហាញពីបម្រើបម្រាស់នៃគេហទំព័រនេះ នាងខ្ញុំសូមលើកយក User Diagram មកបង្ហាញ។

៣.២.១ ការកំណត់មុខងារអ្នកប្រើប្រាស់

នៅក្នុងវេបសាយនេះ នាងខ្ញុំបានកំណត់នូវមុខងារសម្រាប់អ្នកប្រើប្រាស់ដូចខាងក្រោម

- User Registration
- User Login
- Post question (Read, Create)
- Comment (Read, Create)
- Topic (Read, Create)
- Profile Management (view owner posts)

៣.៣ .ការវិភាលើរចនាសម្ព័ន្ធវេបសាយ

ដើម្បីធានាបានថាវេបសាយមួយត្រូវបានរចនាឡើងសមស្របតាមតម្រូវការអ្នកប្រើប្រាស់គេចាំបាច់ណាស់ត្រូធ្វើការវិភាគទៅលើរចនាសម្ព័ន្ធនៃវេបសាយនោះជាមុនសិន។ ដូច្នេះហើយទើបនាងខ្ញុំបានសិក្សា និងបានវិភាគយ៉ាងម៉ត់ចត់ចំពោះដំណើរការនៃវេបសាយទាំងមូលដែលមានដូចតទៅ៖

៣.៣ ១. ការកំណត់ទូទៅ Entities អ្នកប្រើប្រាស់

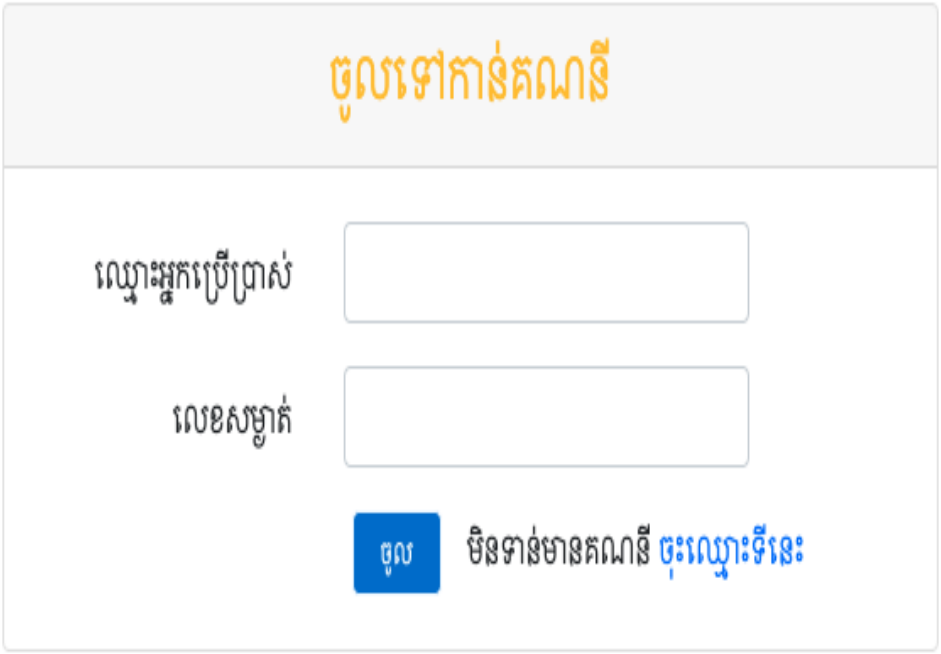
ឆ្លងកាត់ការវិភាគទៅលើវេបសាយយ៉ាងល្អិតល្អន់នាងខ្ញុំបានធ្វើការកំណត់ទិន្នន័យដូចខាងក្រោម៖

- User Registration: អ្នកប្រើប្រាស់ចាំបាច់ត្រូវចុះឈ្មោះបង្កើតគណនីថ្មីតាមរយៈការប្រើប្រាស់ Email។
- User Login: អ្នកប្រើប្រាស់ដែលមានគណនីស្រាប់ធ្វើការចូលទៅកាន់គណនីគាត់តាមរយៈឈ្មោះ និងលេខសម្ងាត់របស់គាត់។

- ការសួរសំណួរ (Post Question): អ្នកប្រើប្រាស់ធ្វើការសួរសំណួរ
- អ្នកប្រើប្រាស់អាចធ្វើការពិនិត្យ និងបញ្ចេញមតិបាន
- អ្នកប្រើប្រាស់អាចបង្កើតប្រធានបទថ្មី
- អ្នកប្រើប្រាស់អាចធ្វើការពិនិត្យពីសកម្មភាពដែលគាត់បានចូលរួមនៅក្នុងវេបសាយទាំងមូល។

៣.៣.២ សម្ងតិកម្ម Entities Page **អ្នកប្រើប្រាស់**

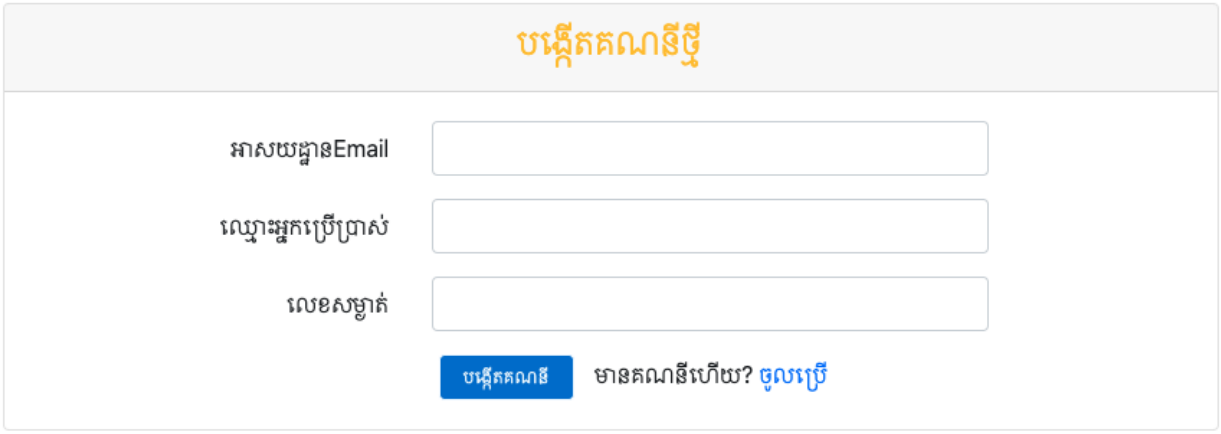
៣.៣. ២.១ Login



រូបភាពទី២0 User Login

៣. ៣.២.២ Register Page

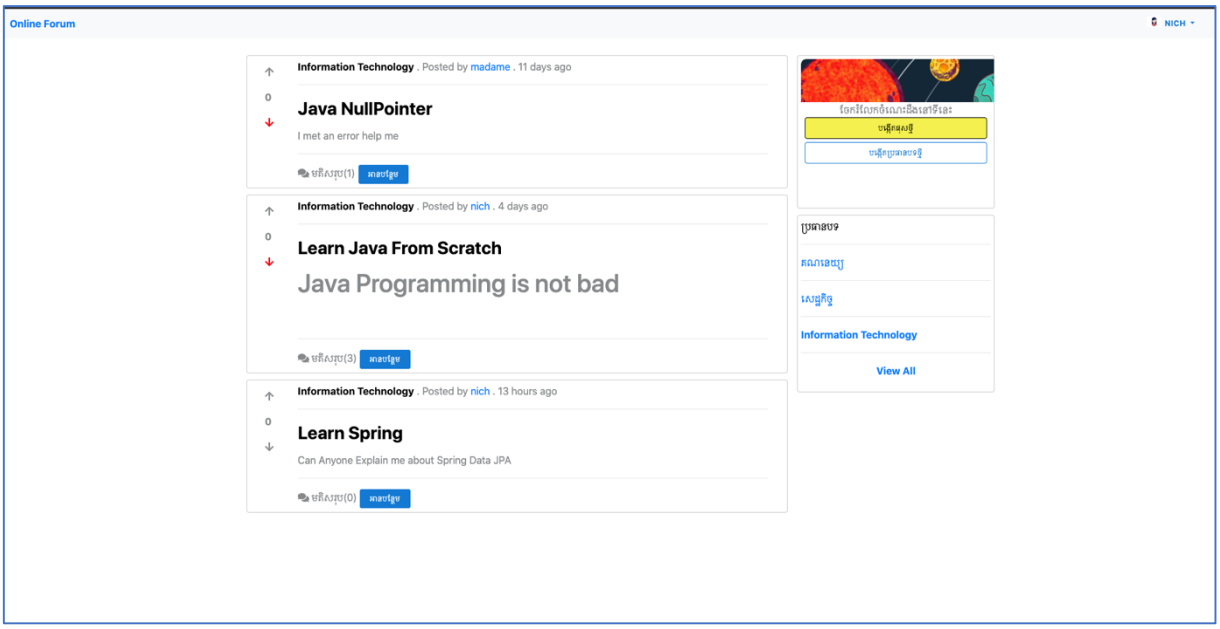
អ្នកប្រើប្រាស់ធ្វើការបង្កើតគណនីដោយប្រើប្រាស់ អ៊ីមែល ឈ្មោះ លេខសម្ងាត់ និងធ្វើការផ្ទៀងផ្ទាត់នូវគណនីរបស់ខ្លួន។ ការធ្វើបែបនេះបង្ការអ្នកប្រើប្រាស់ពីការ ដាក់នូវព័ត៌មានដែលមិនពិតប្រាកដដែលជាហេតុបង្កទៅជាហានិភ័យនានាទៅកាន់អ្នកប្រើប្រាស់ដ៏ទៃទៀតនៅក្នុងប្រព័ន្ធ។



រូបភាពទី២១ User Register

៣.៣.២.៣ Homepage

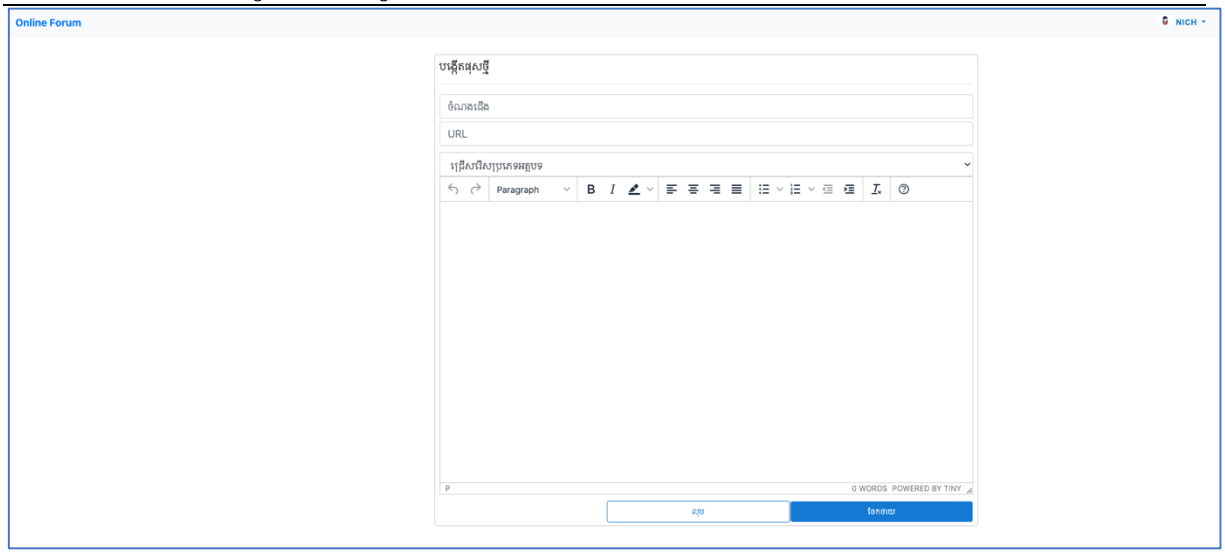
ជាផ្ទាំងដំបូងនៃកម្មវិធី។



រូបភាពទី២២ Homepage

៣. ៣.២.៤. Create Post Page

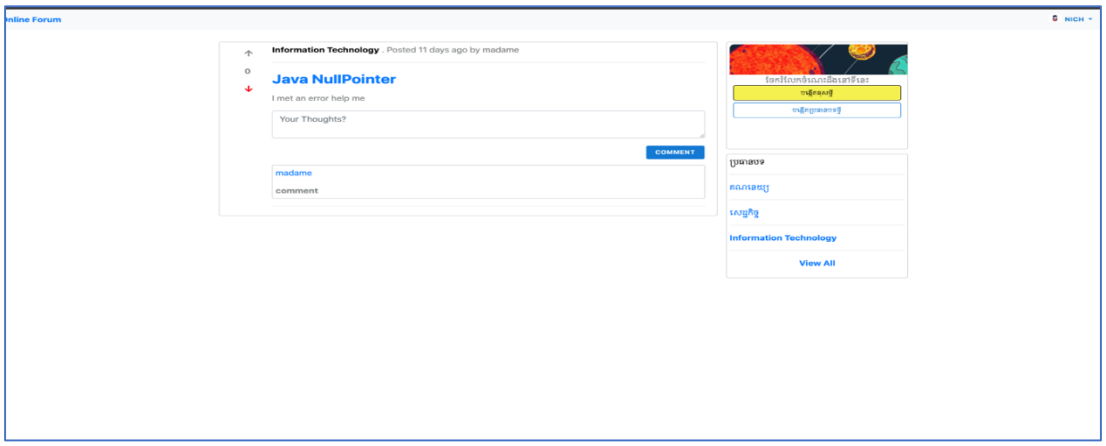
តាមរយៈផ្ទាំងជំនួយ Editor អ្នកប្រើប្រាស់អាចធ្វើការបង្កើតនូវសំណួរថ្មីបាន។ នៅក្នុងគេហទំព័រមួយនេះ នាងខ្ញុំសម្រេចប្រើប្រាស់នូវ Editorមួយដែលអ្នកប្រើប្រាស់អាចធ្វើការ កំណត់នូវទម្រង់អក្សរដែលគាត់ចង់បាន។



រូបភាពទី២៣ Create Post

៣.៣.២.៥. Post Detail

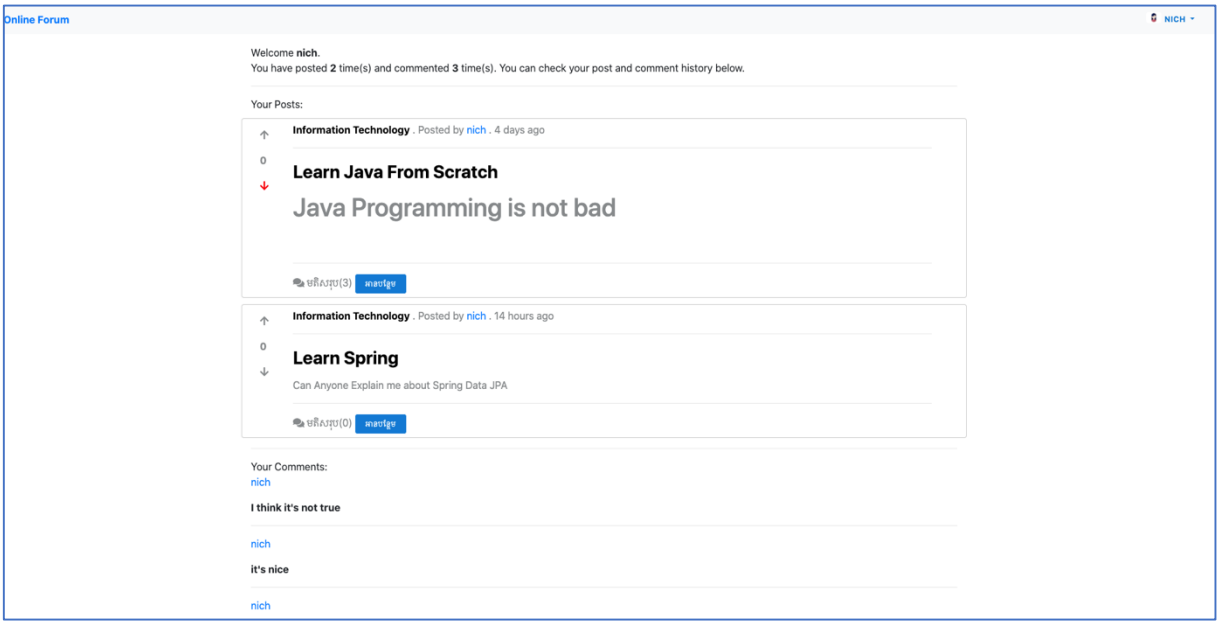
អ្នកប្រើប្រាស់អាចពិនិត្យមើលផុសណាមួយជាក់លាក់ មិនតែប៉ុណ្ណោះគេក៏អាចធ្វើការបោះឆ្នោតនិង បញ្ចេញមតិទៅលើផុសនោះផងដែរ។



រូបភាពទី២៤ Post Detail

៣.៣.២.៦. View Own Profile

អ្នកប្រើប្រាស់អាចពិនិត្យមើលផុសដែលខ្លួនបានបង្កើត និងមតិនានាដែលគាត់បានបញ្ចេញ។



រូបភាពទី២៥ View Own Profile

ជំពូកទី៤
ការវិភាគទិន្នន័យ និងគេហទំព័រ

៤ .១.ការរចនា Database

៤ .១.១.ការកំណត់ឈ្មោះរបស់ Table

ដោយពឹងផ្អែកលើEntity នៅជំពូកទីបី យើងកំណត់បានតារាងខាងក្រោម៖

Entity	Table	Description
User	user	កត់ត្រាពត៌មានអ្នកប្រើប្រាស់
Vote	vote	កត់ត្រាការបោះឆ្នោតលើសំណួរ
Subforum	subforum	កត់ត្រាអំពីប្រធានបទ
Post	post	កត់ត្រាពត៌មានអំពីសំណួរ
Comment	comment	កត់ត្រាពត៌មានអំពីការបញ្ចេញមតិ
VerificationToken	token	កត់ត្រា Token
Refresh Token	refresh_token	កត់ត្រា Refresh Token

តារាងទី៤.១.១ ការកំណត់ឈ្មោះ Table

៤ .១.២.ការពន្យល់អំពី Data Dictionary

Data Dictionary ជាផ្នែកដែលគេប្រើប្រាស់ដើម្បីធ្វើការបកស្រាយលើ Entity និងAttribute ។ដើម្បីផ្តល់ភាពងាយស្រួលដល់ការសិក្សានិងអ្នកអានកាន់តែយល់អំពី តារាងនីមួយៗព្រមទាំងទទួលការប្រើប្រាស់លំអិតអំពីតួនាទីនៃទិន្នន័យដូចខាងក្រោម៖

user		
Entity	Table	Description
id	bigint	
created	date	ថ្ងៃចុះឈ្មោះ
email	varchar(255)	អ៊ីមែល

enabled	boolean	ស្ថានភាព
password	varchar(255)	លេខសម្ងាត់
username	Varchar(255)	ឈ្មោះ

តារាងទី៤.១.២ ពន្យល់អំពី Table User

token		
Entity	Table	Description
id	bigint	
expiry_date	date	ថ្ងៃផុតកំណត់
token	varchar(255)	
user_user_id	boolean	អ្នកប្រើប្រាស់

តារាងទី៤.១.២ ពន្យល់អំពី Table Token

subforum		
Entity	Table	Description
id	bigint	
created_date	date	ថ្ងៃបង្កើត

description	varchar(255)	ពិពណ៌នា
user_user_id	boolean	អ្នកប្រើប្រាស់
name	varchar(255)	ឈ្មោះ

តារាងទី៤.១.២ ពន្យល់អំពី Table subforum

post		
Entity	Table	Description
post_id	bigint	
created_date	date	ថ្ងៃបង្កើត
description	varchar(255)	ពិពណ៌នា
user_id	boolean	អ្នកប្រើប្រាស់
post_name	varchar(255)	ឈ្មោះផុស
url	varchar(255)	តំណ
vote_count	integer	ចំនួនបោះឆ្នោត
id	bigint	ប្រធានបទ

តារាងទី៤.១.២ ពន្យល់អំពី Table post

vote		
Entity	Table	Description

vote_id	bigint	
user_id	bigint	អ្នកប្រើប្រាស់
post_id	bigint	សំណួរ
vote_type	integer	ប្រភេទបោះឆ្នោត

តារាងទី៤.១.២ ពន្យល់អំពី Table vote

comment		
Entity	Table	Description
id	bigint	
created_date	date	ថ្ងៃបង្កើត
text	text	មតិ
user_id	bigint	អ្នកប្រើប្រាស់
post_id	bigint	សំណួរ

តារាងទី៤.១.២ ពន្យល់អំពី Table comment

refresh_token		
Entity	Table	Description
id	bigint	

created_date	date	ថ្ងៃបង្កើត
token	varchar(255)	

តារាងទី៤.២ ពន្យល់អំពី Table refresh_token

៤.២ .ការរចនា លើផ្នាំងវេបសាយ

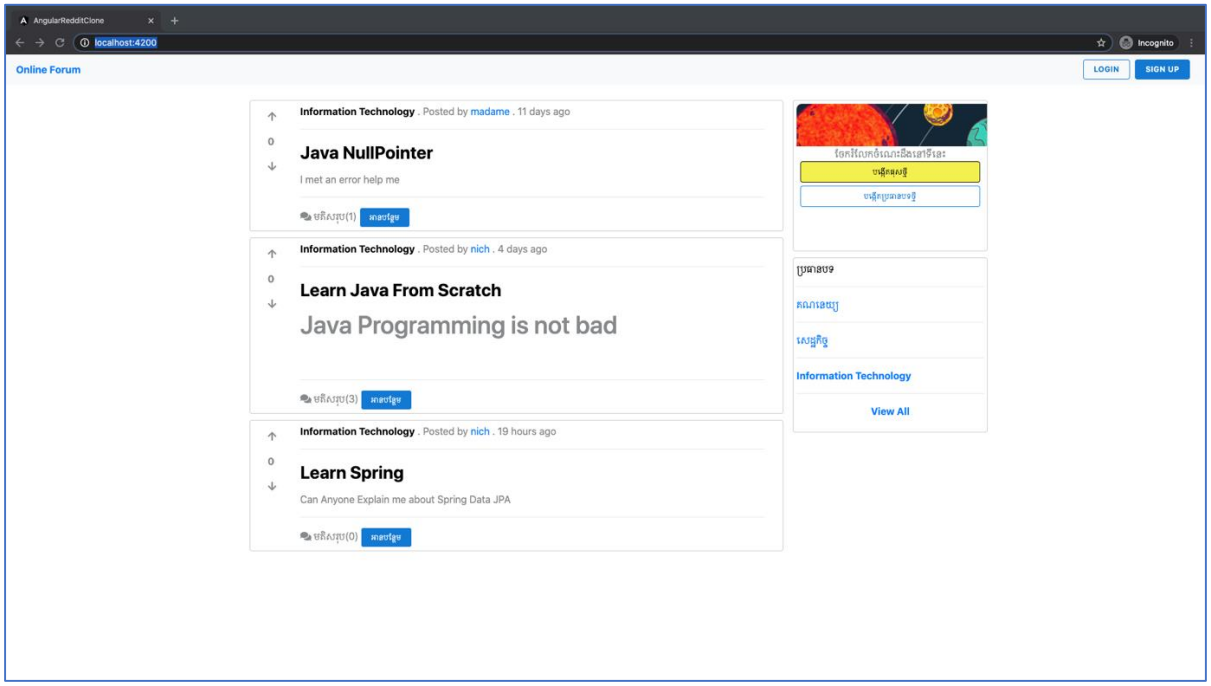
៤.២ .១.រៀបLogin និង Register ទៅកាន់វេបសាយ

ដើម្បីចូលទៅកាន់វេបសាយយើងត្រូវ៖

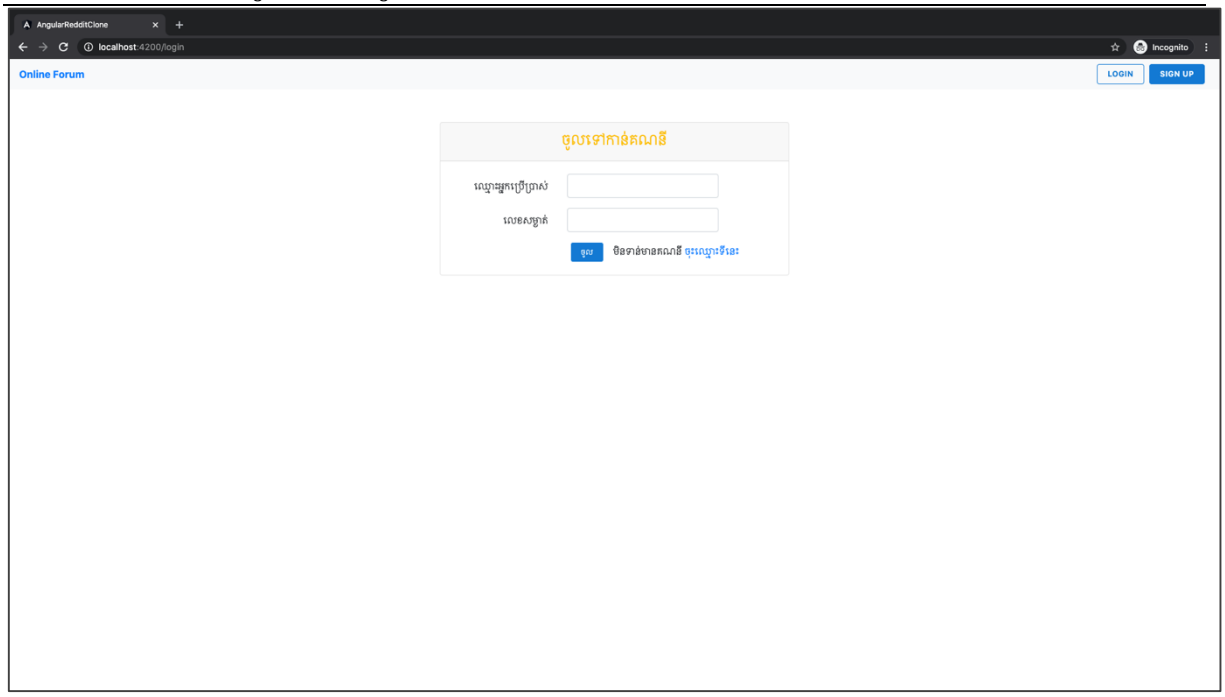
១. បើកBrowser

២. វាយអាសយដ្ឋានរបស់គេហទំព័រ

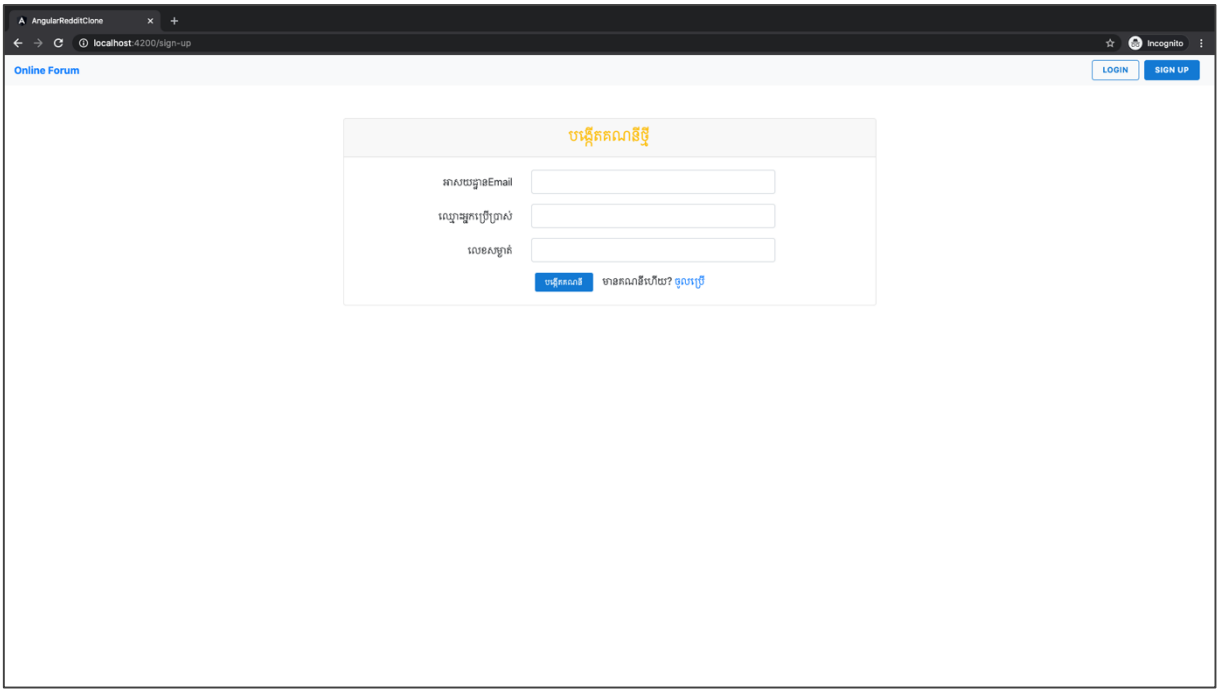
៣. ធ្វើការ Login ឬ Register ដោយប្រើប្រាស់ អ៊ីមែលនិង លេខសម្ងាត់



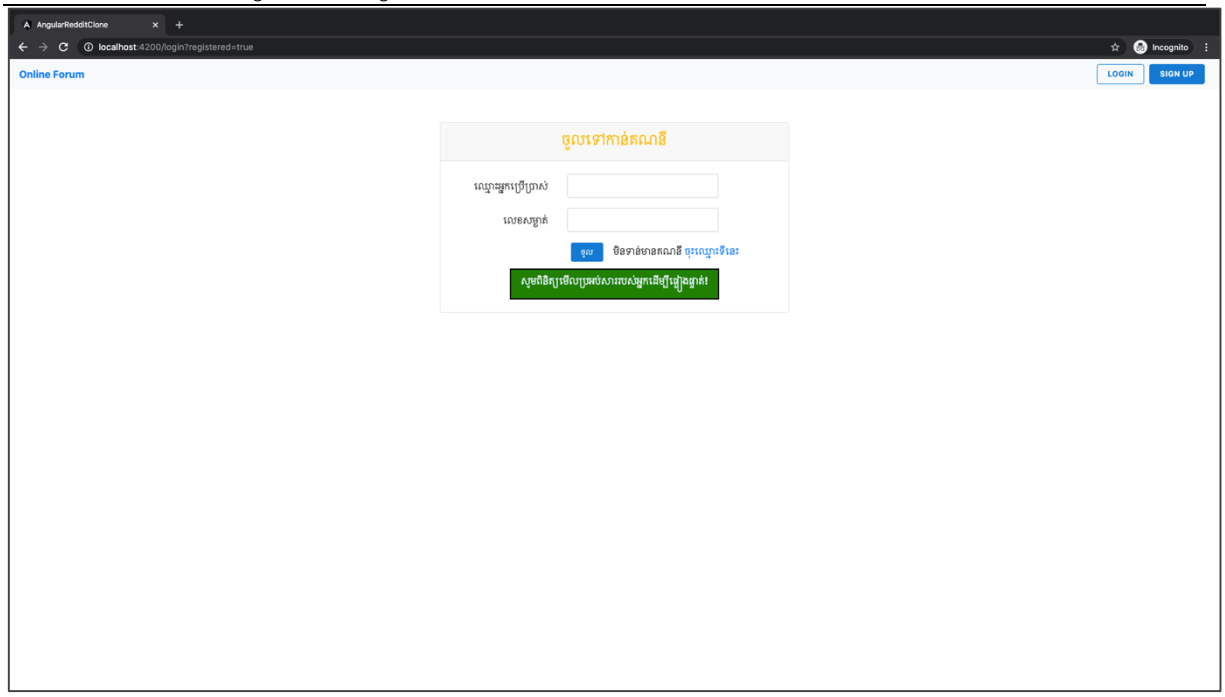
រូបភាពទី២៦ Homepage



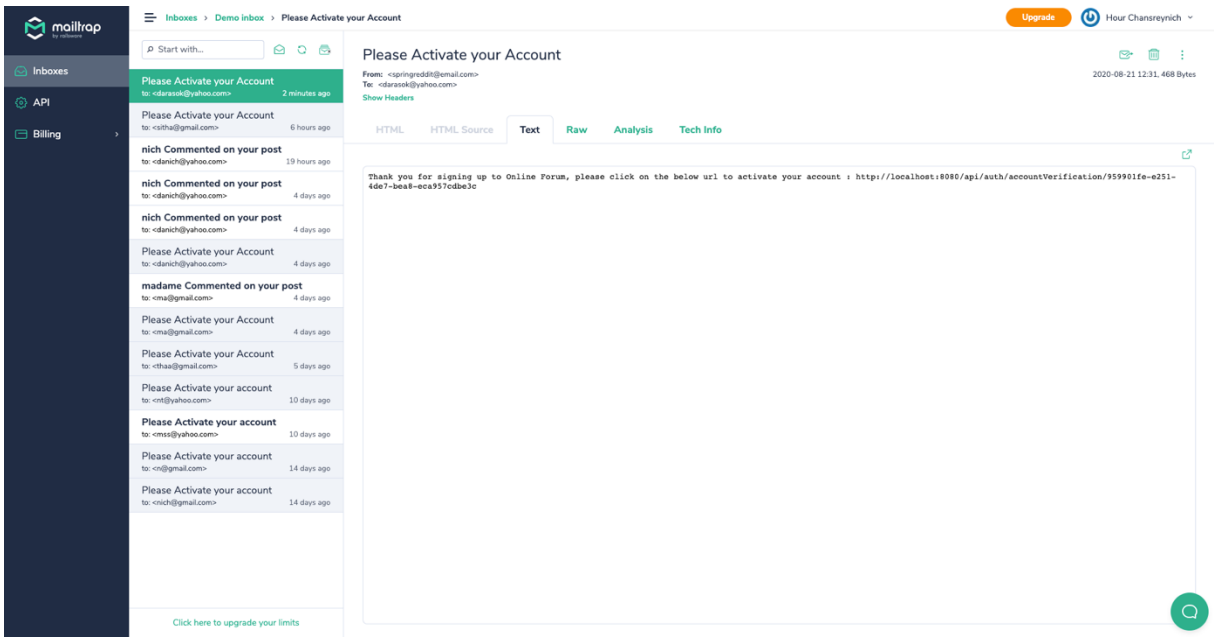
រូបភាពទី២៧ Login Page



រូបភាពទី២៨ Signup Page



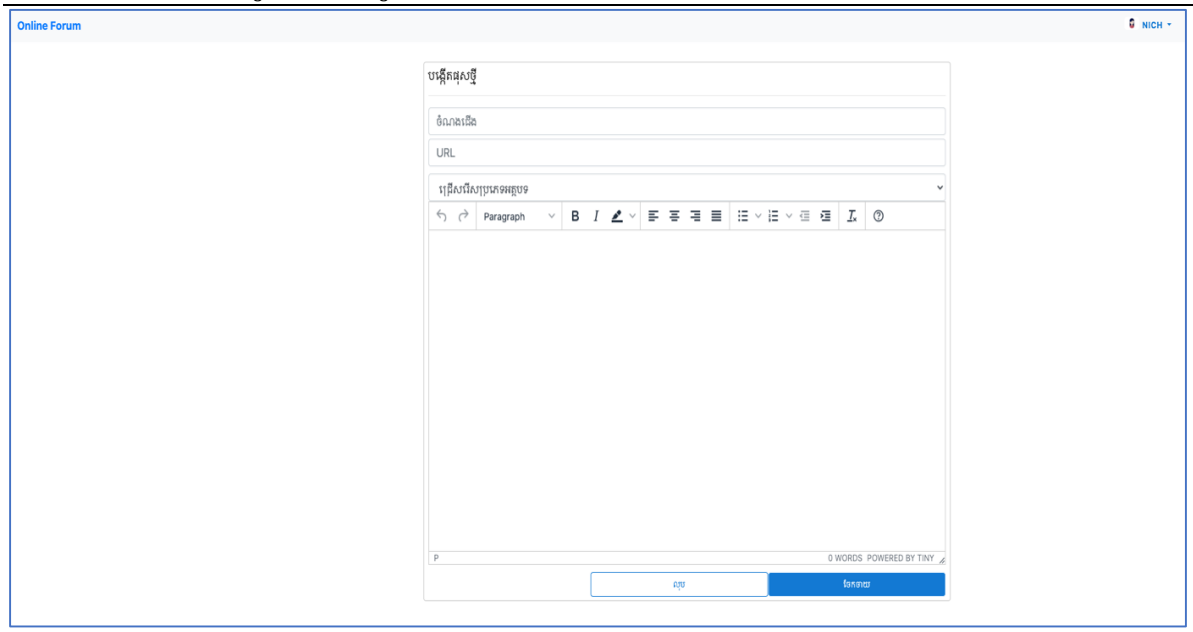
រូបភាពទី២៩ After Signup Page



រូបភាពទី៣០ ផ្ទៀងផ្ទាត់គណនី

៤.២.២ .Create Post

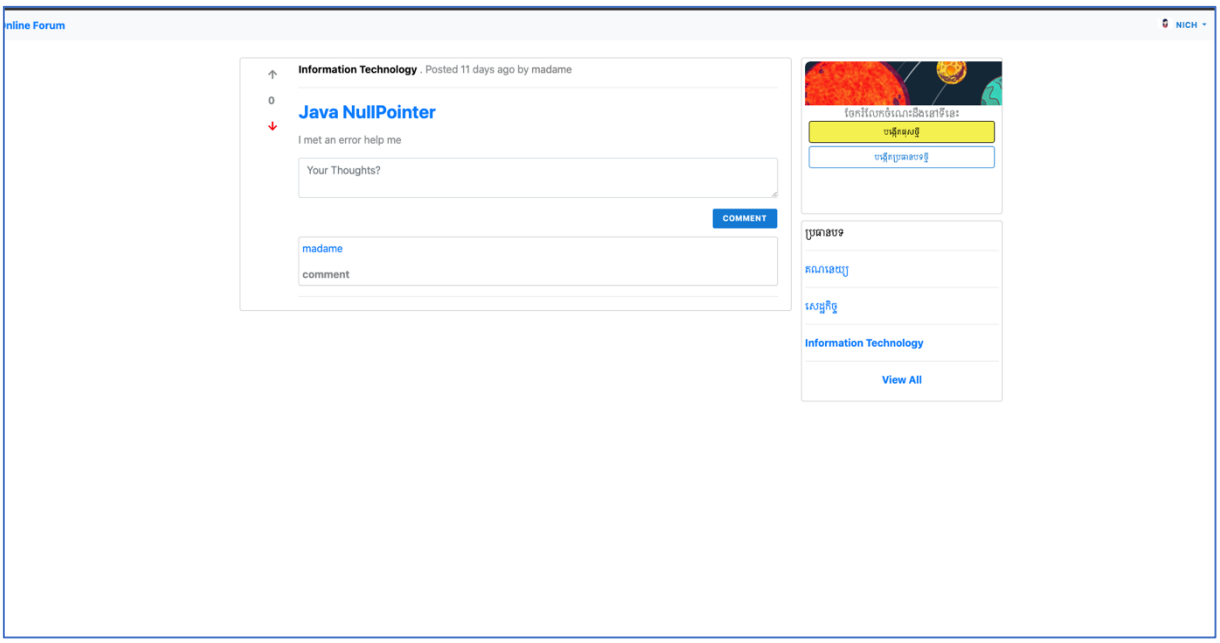
តាមរយៈផ្ទាំងជំនួយ Editor អ្នកប្រើប្រាស់អាចធ្វើការបង្កើតនូវសំណួរថ្មីបាន។



រូបភាពទី៣១ Create Post

៤.២ .៣.Comment and Vote

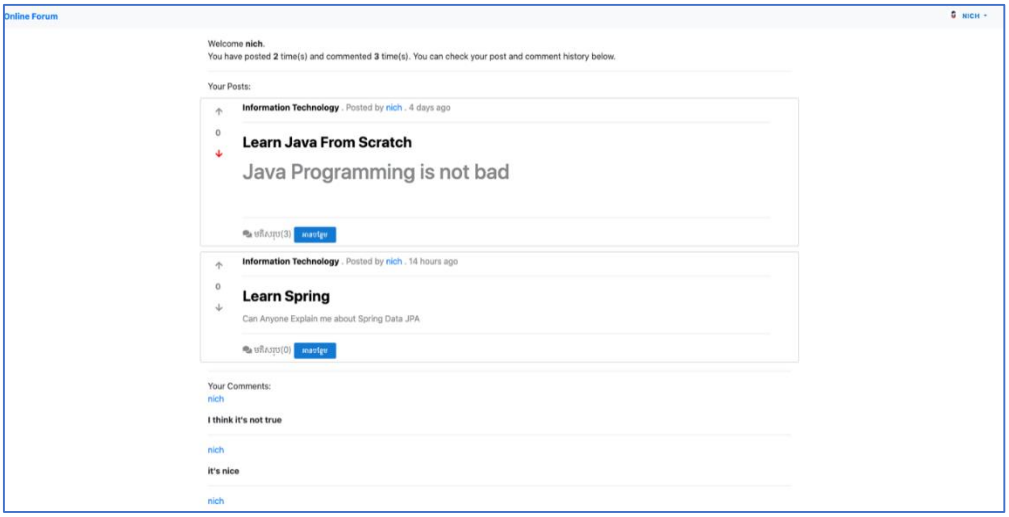
អ្នកប្រើប្រាស់អាចពិនិត្យមើលផុសណាមួយជាក់លាក់មិនតែប៉ុណ្ណោះគេក៏អាចធ្វើការបោះឆ្នោត និងបញ្ចេញមតិទៅលើផុសនោះផងដែរ។



រូបភាពទី៣២ Comment, Vote

៤.២.៤. View Own Profile

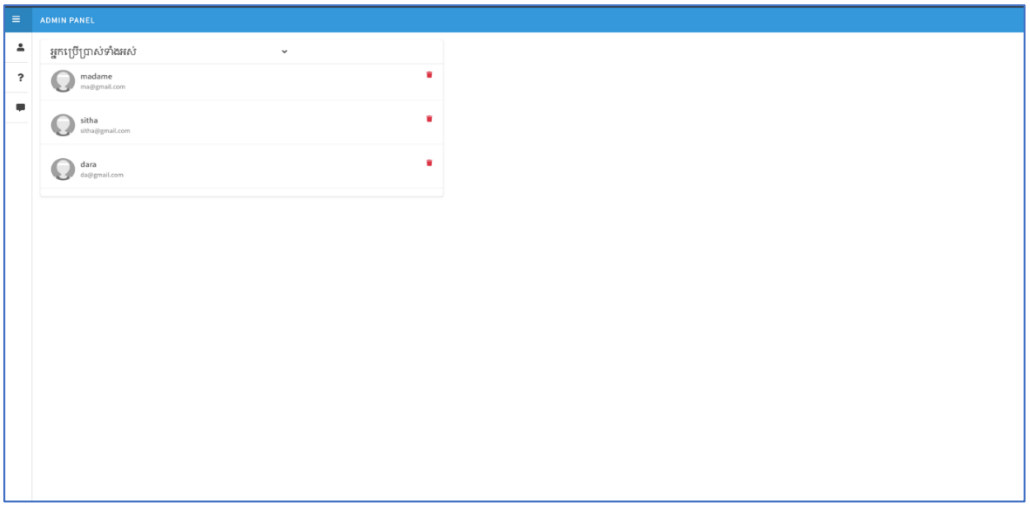
អ្នកប្រើប្រាស់អាចពិនិត្យមើលផុសដែលខ្លួនបានបង្កើត និងមតិនានាដែលគាត់បានបញ្ចេញ។



រូបភាពទី៣៣ View Own Profile

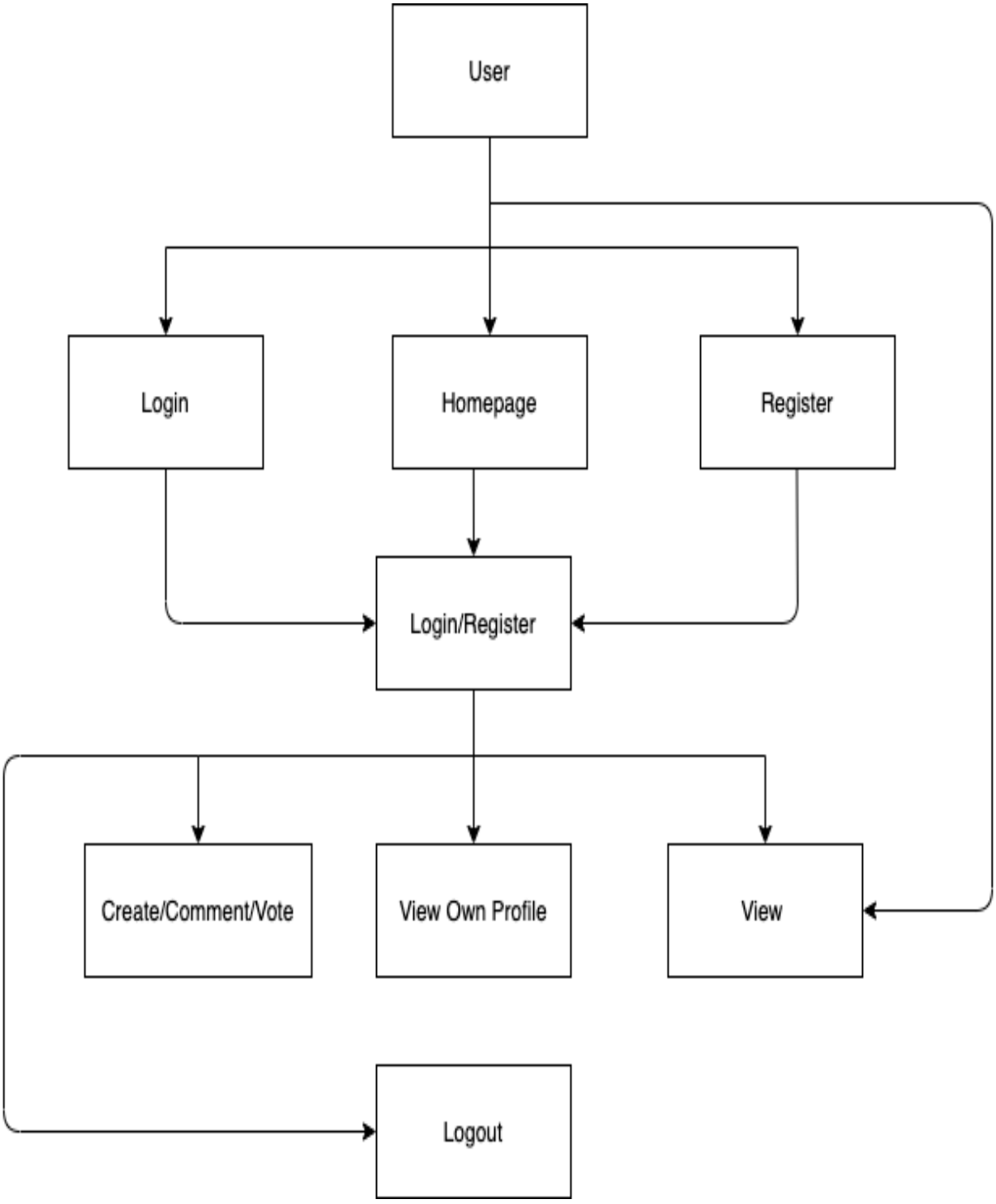
៤.២.៥. Admin Page

Admin មានសិទ្ធិលុបរាល់អ្នកប្រើប្រាស់ទាំងឡាយណាដែលប្រើប្រាស់ខុសពីការកំណត់។



រូបភាពទី៣៤ Admin Page

៤.៣.ដំណើរការគ្រោងទេបសាយ


User
Login
Homepage
Register
Login/Register
Create/Comment/Vote
View Own Profile
View
Logout

សេចក្តីសន្និដ្ឋាន

បន្ទាប់ពីបានឃើញនូវបច្ចេកវិទ្យាបានរីកលូតលាស់គួរអោយកត់សម្គាល់ និងបានធ្វើការសិក្សាស្រាវជ្រាវទៅលើ Online Forum វេបសាយ តាមរយៈការវិភាគទិន្នន័យ និងការសរសេរកូដកន្លងមកនេះជាចុងក្រោយយើងទទួលបានគេហទំព័រមួយដែលអនុញ្ញាតអោយអ្នកប្រើប្រាស់អាចចែករំលែកចំណេះដឹងទៅវិញទៅមកបាន។ តាមរយៈវេបសាយនេះយើងសង្ឃឹមថាអ្នកប្រើប្រាស់ជាពិសេសសិស្សានុសិស្សអាចប្រើប្រាស់វាជា ទីកន្លែងសម្រាប់ចែករំលែកចំណេះដឹងគ្នាទៅវិញទៅមក។ ទោះបីជាភាពជោគជ័យនេះអាចសម្រួលដល់ការប្រើប្រាស់យ៉ាងណាក៏ដោយ យើងនៅតែមានចំណុចខ្វះខាតមួយចំនួនដែលជៀសពុំបាន ដោយហេតុថាពេលវេលាមានកំណត់សម្រាប់ដំណើរការស្រាវជ្រាវមួយនេះ។ ហេតុនេះនាងខ្ញុំនឹងរង់ចាំទទួលនូវការរិះគន់ស្ថាបនា ប្រកបដោយភាពរីករាយពីគ្រប់មជ្ឈដ្ឋាន ក្នុងគោលបំណងធ្វើការអភិវឌ្ឍន៍គម្រោងមួយនេះអោយមានភាពកាន់តែល្អប្រសើរ។

ឯកសារយោង

1. https://angular.io/guide/architecture
2. https://spring.io/
3. https://www.techiediaries.com/observables-and-subjects-tutorial/
4. https://programmingtechie.com/
5. https://www.postgresql.org/docs/9.1/sql-createfunction.html

ឧបសម្ព័ន្ធ

ក្បួនក្នុងការបង្កើតកម្មវិធី

- Spring Project
- DataRestConfig.java

```
@Configuration
public class DataRestConfig implements RepositoryRestConfigurer {
    @Override
    public void
configureRepositoryRestConfiguration(RepositoryRestConfiguration config)
{
        config.exposeIdsFor(User.class);
        config.setBasePath("/api");
    }
}
```

- SecurityConfig.java

```
@Configuration
@EnableWebSecurity
@AllArgsConstructor
public class SecurityConfiguration extends WebSecurityConfigurerAdapter
{

    private  final UserDetailsService userDetailsService;
    private final JwtAuthenticationFilter jwtAuthenticationFilter;

    @Bean(BeanIds.AUTHENTICATION_MANAGER)
    @Override
    public AuthenticationManager authenticationManagerBean() throws
Exception {
        return super.authenticationManagerBean();
    }


    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.csrf().disable()
                .authorizeRequests()
                .antMatchers("/api/auth/**")
                .permitAll()
                .anyRequest().permitAll();
//        http.addFilterBefore(jwtAuthenticationFilter,
UsernamePasswordAuthenticationFilter.class);
    }
```

```
    public void configureGlobal(AuthenticationManagerBuilder
authenticationManagerBuilder) throws Exception{

authenticationManagerBuilder.userDetailsService(userDetailsService).pass
wordEncoder(passwordEncoder());
    }

    @Bean
    PasswordEncoder passwordEncoder(){
        return new BCryptPasswordEncoder();
    }
}
```

- WebConfig.java

```
@Configuration
@EnableWebMvc
public class WebConfig  implements WebMvcConfigurer {
    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**")
                .allowedOrigins("*")
                .allowedMethods("*")
                .maxAge(3600L)
                .allowedHeaders("*")
                .exposedHeaders("Authorization")
                .allowCredentials(true);
    }
}
```

- AuthenticationResponse.java

```
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.time.Instant;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class AuthenticationResponse {
    private String authenticationToken;
    private String refreshToken;
```

```
    private String username;
    private Instant expiresAt;
}
```

- LoginRequest.java

```
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class LoginRequest {
    private String username;
    private String password;
}
```

- RefreshTokenRequest.java

```
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.validation.constraints.NotBlank;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class RefreshTokenRequest {
    @NotBlank
    private String refreshToken;
    private String username;
}
```

- RegisterRequest.java

- ```
  import lombok.AllArgsConstructor;
  import lombok.Data;
  import lombok.NoArgsConstructor;

  @Data
  @AllArgsConstructor
  @NoArgsConstructor
  public class RegisterRequest {
      private String username;
      private String email;
      private String password;
  }
  ```
  ```

- JwtAuthenticationFilter.java

```
import com.example.demo.service.JwtProvider;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToke
n;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import
org.springframework.security.web.authentication.WebAuthenticationDetailsSource
;
import org.springframework.stereotype.Component;
import org.springframework.util.StringUtils;
import org.springframework.web.filter.OncePerRequestFilter;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@Component
public class JwtAuthenticationFilter extends OncePerRequestFilter {

    @Autowired
    private JwtProvider jwtProvider;
    @Qualifier("userDetailServiceImp")
    @Autowired
    private UserDetailsService userDetailsService;

    @Override
    protected void doFilterInternal(HttpServletRequest request,
HttpServletResponse response,
                                    FilterChain filterChain) throws
ServletException, IOException {
        String jwt = getJwtFromRequest(request);

        if (StringUtils.hasText(jwt)) {
            String username = jwtProvider.getUsernameFromJwt(jwt);

            UserDetails userDetails =
userDetailsService.loadUserByUsername(username);
            UsernamePasswordAuthenticationToken authentication = new
UsernamePasswordAuthenticationToken(userDetails,
                    null, userDetails.getAuthorities());
            authentication.setDetails(new
WebAuthenticationDetailsSource().buildDetails(request));


SecurityContextHolder.getContext().setAuthentication(authentication);
        }
```

```
        filterChain.doFilter(request, response);
    }

    private String getJwtFromRequest(HttpServletRequest request) {
        String bearerToken = request.getHeader("Authorization");

        if (StringUtils.hasText(bearerToken) && bearerToken.startsWith("Bearer
")) {
            return bearerToken.substring(7);
        }
        return bearerToken;
    }
}
```

- AuthController.java

```
import com.example.demo.dto.AuthenticationResponse;
import com.example.demo.dto.LoginRequest;
import com.example.demo.dto.RegisterRequest;
import com.example.demo.model.RefreshToken;
import com.example.demo.service.AuthService;
import com.example.demo.service.RefreshTokenService;
import lombok.AllArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;

@RestController
@RequestMapping("/api/auth")
@AllArgsConstructor
public class AuthController {
    @Autowired
    private final AuthService authService;
    private final RefreshTokenService refreshTokenService;
    @PostMapping("/signup")
    public ResponseEntity sigup(@RequestBody RegisterRequest
registerRequest){
        authService.signup(registerRequest);
        return new ResponseEntity("User Register Successful, Check your
email",HttpStatus.OK);
    }
    @PostMapping("/login")
    public AuthenticationResponse login(@RequestBody LoginRequest
loginRequest) {
        System.out.println("Login success");
        return authService.login(loginRequest);
    }
    @GetMapping("accountVerification/{token}")
    public ResponseEntity<String> verifyAccount(@PathVariable String
```

```
token){
        authService.verifyAccount(token);
        return new ResponseEntity<>("Account Activated
Successfully",HttpStatus.OK);
    }
    @PostMapping("/logout")
    public ResponseEntity<String> logout(@Valid @RequestBody
RefreshToken refreshToken) {
        refreshTokenService.deleteRefreshToken(refreshToken.getToken());
        return ResponseEntity.status(HttpStatus.OK).body("Refresh Token
Deleted Successfully!!");
    }

    @GetMapping("/works")
    public String work(){
        return "worked";
    }
}
```

- AuthService.java

```
import com.example.demo.dto.AuthenticationResponse;
import com.example.demo.dto.LoginRequest;
import com.example.demo.dto.RefreshTokenRequest;
import com.example.demo.dto.RegisterRequest;
import com.example.demo.exception.SpringForumException;
import com.example.demo.model.NotificationEmail;
import com.example.demo.model.User;
import com.example.demo.model.VerificationToken;
import com.example.demo.repository.UserRepository;
import com.example.demo.repository.VerificationTokenRepository;
import lombok.AllArgsConstructor;
import
org.springframework.security.authentication.AnonymousAuthenticationToken
;
import
org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.authentication.UsernamePasswordAuthenticati
onToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import
org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.time.Instant;
import java.util.Optional;
import java.util.UUID;
```

```
@Service
@AllArgsConstructor
@Transactional
public class AuthService {

    private final PasswordEncoder passwordEncoder;
    private final UserRepository userRepository;
    private final VerificationTokenRepository
verificationTokenRepository;
    private final MailService mailService;
    private final AuthenticationManager authenticationManager;
    private final JwtProvider jwtProvider;
    private final RefreshTokenService refreshTokenService;

    public void signup(RegisterRequest registerRequest) {
        User user = new User();
        user.setUsername(registerRequest.getUsername());
        user.setEmail(registerRequest.getEmail());

user.setPassword(passwordEncoder.encode(registerRequest.getPassword()));
        user.setCreated(Instant.now());
        user.setEnabled(false);

        userRepository.save(user);


    @Transactional(readOnly = true)
    public User getCurrentUser() {
        org.springframework.security.core.userdetails.User principal =
(org.springframework.security.core.userdetails.User)
SecurityContextHolder.
                getContext().getAuthentication().getPrincipal();
        return userRepository.findByUsername(principal.getUsername())
                .orElseThrow(() -> new UsernameNotFoundException("User
name not found - " + principal.getUsername()));
    }

    private void fetchUserAndEnable(VerificationToken verificationToken)
{
        String username = verificationToken.getUser().getUsername();
        User user =
userRepository.findByUsername(username).orElseThrow(() -> new
SpringForumException("User not found with name - " + username));
        user.setEnabled(true);
        userRepository.save(user);
    }

    private String generateVerificationToken(User user) {
        String token = UUID.randomUUID().toString();
        VerificationToken verificationToken = new VerificationToken();
        verificationToken.setToken(token);
        verificationToken.setUser(user);
```

```
        verificationTokenRepository.save(verificationToken);
        return token;
    }

    public void verifyAccount(String token) {
        Optional<VerificationToken> verificationToken =
verificationTokenRepository.findByToken(token);
        fetchUserAndEnable(verificationToken.orElseThrow(() -> new
SpringForumException("Invalid Token")));
    }

    public AuthenticationResponse login(LoginRequest loginRequest) {
        Authentication authenticate =
authenticationManager.authenticate(new
UsernamePasswordAuthenticationToken(loginRequest.getUsername(),
                loginRequest.getPassword()));

SecurityContextHolder.getContext().setAuthentication(authenticate);
        String token = jwtProvider.generateToken(authenticate);
        return new AuthenticationResponse().builder()
                .authenticationToken(token)

.refreshToken(refreshTokenService.generateRefreshToken().getToken())

.expiresAt(Instant.now().plusMillis(jwtProvider.getJwtExpirationInMillis
()))
                .username(loginRequest.getUsername())
                .build();
    }

    public AuthenticationResponse refreshToken(RefreshTokenRequest
refreshTokenRequest) {

refreshTokenService.validateRefreshToken(refreshTokenRequest.getRefreshT
oken());
        String token =
jwtProvider.generateTokenWithUserName(refreshTokenRequest.getUsername())
;
        return AuthenticationResponse.builder()
                .authenticationToken(token)
                .refreshToken(refreshTokenRequest.getRefreshToken())

.expiresAt(Instant.now().plusMillis(jwtProvider.getJwtExpirationInMillis
()))
                .username(refreshTokenRequest.getUsername())
                .build();
    }

    public boolean isLoggedIn() {
        Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();
        return !(authentication instanceof AnonymousAuthenticationToken)
&& authentication.isAuthenticated();
```

```
        }
    }
```

- JwtProvider.java

```
import com.example.demo.exception.SpringForumException;
import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.userdetails.User;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import javax.xml.crypto.Data;
import java.io.IOException;
import java.io.InputStream;
import java.security.*;
import java.security.cert.CertificateException;
import java.time.Instant;
import java.util.Date;

import static io.jsonwebtoken.Jwts.parser;
import static java.util.Date.from;

@Service
public class JwtProvider {
    private KeyStore keyStore;
    @Value("${jwt.expiration.time}")
    private Long jwtExpirationInMillis;

    @PostConstruct
    public void init() {
        try {
            keyStore = KeyStore.getInstance("JKS");
            InputStream resourceAsStream =
getClass().getResourceAsStream("/springblog.jks");
            keyStore.load(resourceAsStream, "secret".toCharArray());
        } catch (KeyStoreException | CertificateException |
NoSuchAlgorithmException | IOException e) {
            throw new SpringForumException("Exception occurred while loading
keystore");
        }

    }

    public String generateToken(Authentication authentication) {
        org.springframework.security.core.userdetails.User principal = (User)
authentication.getPrincipal();
        return Jwts.builder()
                .setSubject(principal.getUsername())
```

```
                .setIssuedAt(from(Instant.now()))
                .signWith(getPrivateKey())

.setExpiration(from(Instant.now().plusMillis(jwtExpirationInMillis)))
                .compact();
    }

    private PrivateKey getPrivateKey() {
        try {
            return (PrivateKey) keyStore.getKey("springblog",
"secret".toCharArray());
        } catch (KeyStoreException | NoSuchAlgorithmException |
UnrecoverableKeyException e) {
            throw new SpringForumException("Exception occured while retrieving
public key from keystore");
        }
    }

    public String generateTokenWithUserName(String username) {
        return Jwts.builder()
                .setSubject(username)
                .setIssuedAt(from(Instant.now()))
                .signWith(getPrivateKey())

.setExpiration(from(Instant.now().plusMillis(jwtExpirationInMillis)))
                .compact();
    }
    public boolean validateToken(String jwt) {
        parser().setSigningKey(getPublickey()).parseClaimsJws(jwt);
        return true;
    }

    private PublicKey getPublickey() {
        try {
            return keyStore.getCertificate("springblog").getPublicKey();
        } catch (KeyStoreException e) {
            throw new SpringForumException("Exception occured while retrieving
public key from keystore");
        }
    }

    public String getUsernameFromJwt(String token) {
        Claims claims = parser()
                .setSigningKey(getPublickey())
                .parseClaimsJws(token)
                .getBody();

        return claims.getSubject();
    }
    public Long getJwtExpirationInMillis() {
        return jwtExpirationInMillis;
    }
}
```

- UserDetailServiceImp.java

- ```
  import com.example.demo.model.User;
  import com.example.demo.repository.UserRepository;
  import lombok.AllArgsConstructor;
  import org.springframework.security.core.GrantedAuthority;
  import
  org.springframework.security.core.authority.SimpleGrantedAuthority;
  import org.springframework.security.core.userdetails.UserDetails;
  import org.springframework.security.core.userdetails.UserDetailsService;
  import
  org.springframework.security.core.userdetails.UsernameNotFoundException;
  import org.springframework.stereotype.Service;
  import org.springframework.transaction.annotation.Transactional;

  import java.util.Collection;
  import java.util.Optional;

  import static java.util.Collections.singletonList;

  @Service
  @AllArgsConstructor
  public class UserDetailServiceImp implements UserDetailsService {
        private   final UserRepository userRepository;
      @Override
      @Transactional(readOnly = true)
      public UserDetails loadUserByUsername(String username) throws
  UsernameNotFoundException {
          Optional<User> userOptional =
  userRepository.findByUsername(username);
          User user = userOptional
                  .orElseThrow(() -> new UsernameNotFoundException("No
  user " +
                          "Found with username : " + username));

          return new org.springframework.security
                  .core.userdetails.User(user.getUsername(),
  user.getPassword(),
                  user.isEnabled(), true, true,
                  true, getAuthorities("USER"));
      }
      private Collection<? extends GrantedAuthority> getAuthorities(String
  role) {
          return singletonList(new SimpleGrantedAuthority(role));
      }
  }
  ```

- CommentProjection.java
  ```

```
import com.example.demo.model.Comment;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.data.rest.core.config.Projection;

import java.time.Instant;
import java.util.Date;

@Projection(types = Comment.class)
public interface CommentProjection {
    Integer getId();
    @Value("#{target.post.id}")
    Integer getPostId();
    Instant getCreatedDate();
    String getText();
    @Value("#{target.user.username}")
    String getUsername();
}
```

- PostProjection.java

```
import com.example.demo.model.Post;
import com.example.demo.model.Subforum;
import com.example.demo.model.User;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.data.rest.core.config.Projection;

import java.time.Instant;

@Projection(name = "PostProejection",types = {Post.class})
public interface PostProjection {
    Integer getId();
    String getPostName();
    String getUrl();
    String getDescription();
    Integer getVoteCount();
    @Value("#{target.subforum.name}")
    String getSubforumName();
    @Value("#{target.user.username}")
    String getUserName();
    @Value("#{target.comments.size()}")
     Integer getCommentCount();
  Instant getCreatedDate();



}
```

- voteProjection.java

```
import com.example.demo.model.Vote;
import com.example.demo.model.VoteType;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.data.rest.core.config.Projection;

@Projection(types = Vote.class)
public interface VoteProjection {
    VoteType getVoteType();
    @Value("#{target.post.id}")
    Integer getPostId();

}
```

- commentService.java

```
@Service
@AllArgsConstructor
public class CommentService {
    private static final String POST_URL = "";
    private final PostRepository postRepository;
    private final UserRepository userRepository;
    private final AuthService authService;
    private final CommentMapper commentMapper;
    private final CommentRepository commentRepository;
    private final MailContentBuilder mailContentBuilder;
    private final MailService mailService;

    public void save(CommentsDto commentsDto) {
        Post post = postRepository.findById(commentsDto.getPostId())
                .orElseThrow(() -> new
PostNotFoundException(commentsDto.getPostId().toString()));
        Comment comment = commentMapper.map(commentsDto, post,
authService.getCurrentUser());
        commentRepository.save(comment);

        String message =
mailContentBuilder.build(post.getUser().getUsername() + " posted a
comment on your post." + POST_URL);
        sendCommentNotification(message, post.getUser());
    }

    private void sendCommentNotification(String message, User user) {
        mailService.sendMail(new NotificationEmail(user.getUsername() +
" Commented on your post", user.getEmail(), message));
    }

    public List<CommentsDto> getAllCommentsForPost(Long postId) {
        Post post = postRepository.findById(postId).orElseThrow(() ->
new PostNotFoundException(postId.toString()));
```

```
            return commentRepository.findByPost(post)
                    .stream()
                    .map(commentMapper::mapToDto).collect(toList());
        }

        public List<CommentsDto> getAllCommentsForUser(String userName) {
            User user = userRepository.findByUsername(userName)
                    .orElseThrow(() -> new
    UsernameNotFoundException(userName));
            return commentRepository.findAllByUser(user)
                    .stream()
                    .map(commentMapper::mapToDto)
                    .collect(toList());
        }
    }
```

- jwtProvider.java

```
import
com.programming.techie.springredditclone.exceptions.SpringRedditException;
import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.userdetails.User;
import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
import java.io.IOException;
import java.io.InputStream;
import java.security.*;
import java.security.cert.CertificateException;
import java.sql.Date;
import java.time.Instant;

import static io.jsonwebtoken.Jwts.parser;
import static java.util.Date.from;

@Service
public class JwtProvider {

    private KeyStore keyStore;
    @Value("${jwt.expiration.time}")
    private Long jwtExpirationInMillis;

    @PostConstruct
    public void init() {
        try {
            keyStore = KeyStore.getInstance("JKS");
            InputStream resourceAsStream =
getClass().getResourceAsStream("/springblog.jks");
```

```
            keyStore.load(resourceAsStream, "secret".toCharArray());
        } catch (KeyStoreException | CertificateException |
NoSuchAlgorithmException | IOException e) {
            throw new SpringRedditException("Exception occurred while loading
keystore", e);
        }

    }

    public String generateToken(Authentication authentication) {
        User principal = (User) authentication.getPrincipal();
        return Jwts.builder()
                .setSubject(principal.getUsername())
                .setIssuedAt(from(Instant.now()))
                .signWith(getPrivateKey())

.setExpiration(Date.from(Instant.now().plusMillis(jwtExpirationInMillis)))
                .compact();
    }

    public String generateTokenWithUserName(String username) {
        return Jwts.builder()
                .setSubject(username)
                .setIssuedAt(from(Instant.now()))
                .signWith(getPrivateKey())

.setExpiration(Date.from(Instant.now().plusMillis(jwtExpirationInMillis)))
                .compact();
    }

    private PrivateKey getPrivateKey() {
        try {
            return (PrivateKey) keyStore.getKey("springblog",
"secret".toCharArray());
        } catch (KeyStoreException | NoSuchAlgorithmException |
UnrecoverableKeyException e) {
            throw new SpringRedditException("Exception occured while
retrieving public key from keystore", e);
        }
    }

    public boolean validateToken(String jwt) {
        parser().setSigningKey(getPublickey()).parseClaimsJws(jwt);
        return true;
    }

    private PublicKey getPublickey() {
        try {
            return keyStore.getCertificate("springblog").getPublicKey();
        } catch (KeyStoreException e) {
            throw new SpringRedditException("Exception occured while " +
                    "retrieving public key from keystore", e);
        }
    }
```

```
    public String getUsernameFromJwt(String token) {
        Claims claims = parser()
                .setSigningKey(getPublickey())
                .parseClaimsJws(token)
                .getBody();

        return claims.getSubject();
    }

    public Long getJwtExpirationInMillis() {
        return jwtExpirationInMillis;
    }
}
```

. pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.3.2.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>SocialBackend</name>
    <description>Demo project for Spring Boot</description>

    <properties>

        <java.version>1.8</java.version>
        <org.mapstruct.version>1.3.1.Final</org.mapstruct.version>
    </properties>

    <dependencies>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-rest</artifactId>
            <version>2.3.1.RELEASE</version>
        </dependency>
```

```
        <!-- JWT related dependencies-->
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt-api</artifactId>
            <version>0.10.5</version>
        </dependency>
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt-impl</artifactId>
            <scope>runtime</scope>
            <version>0.10.5</version>
        </dependency>
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt-jackson</artifactId>
            <scope>runtime</scope>
            <version>0.10.5</version>
        </dependency>
        <!-- For Displaying time as Relative Time Ago ("Posted 1 Day ago"),
         as this is a Kotlin library, we also need Kotlin runtime libraries-->
        <dependency>
            <groupId>com.github.marlonlom</groupId>
            <artifactId>timeago</artifactId>
            <version>4.0.1</version>
        </dependency>
        <dependency>
            <groupId>org.jetbrains.kotlin</groupId>
            <artifactId>kotlin-stdlib-jdk8</artifactId>
            <version>${kotlin.version}</version>
        </dependency>
        <dependency>
            <groupId>org.jetbrains.kotlin</groupId>
            <artifactId>kotlin-test</artifactId>
            <version>${kotlin.version}</version>
            <scope>test</scope>
        </dependency>
        <!--
https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-
starter-data-jpa -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
            <version>2.3.1.RELEASE</version>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-mail</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
        <dependency>
```

```
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-jdbc</artifactId>
    </dependency>
    <dependency>
        <groupId>org.postgresql</groupId>
        <artifactId>postgresql</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
        <exclusions>
            <exclusion>
                <groupId>org.junit.vintage</groupId>
                <artifactId>junit-vintage-engine</artifactId>
            </exclusion>
        </exclusions>
    </dependency>
    <dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-test</artifactId>
        <scope>test</scope>
    </dependency>

    <dependency>
        <groupId>org.hibernate.validator</groupId>
        <artifactId>hibernate-validator</artifactId>
        <version>6.0.19.Final</version>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
```

```
    </build>
</project>
```

- Angular Project
    - login-request.payload.ts

```
export interface LoginRequestPayload {
    username: string;
    password: string;
}
```

    - admin.component.html

```
<div class="header">
  <a href="#" id="menu-action">
    <i class="fa fa-bars"></i>
    <span>Close</span>
  </a>
  <div class="logo">
    Admin Panel
  </div>
</div>
<div class="sidebar">
  <ul>
    <li><a href="#"><i class="fa fa-user" aria-hidden="true"></i><span>អ្នកប្រើប្រាស់</span></a></li>
    <li><a href="#"><i class="fa fa-question" aria-hidden="true"></i><span>សំណួរទាំងអស់</span></a></li>
    <li><a href="#"><i class="fa fa-commenting" aria-hidden="true"></i><span>មតិទាំងអស់</span></a></li>
  </ul>
</div>

<!-- Content -->
<div class="main">
  <div class="row">
    <div class="col-lg-5">
      <app-list-user></app-list-user>

    </div>
  </div>

</div>
```

```
<div class="container">


  </div>
```

- admin.component.ts

```
import { Component, OnInit } from '@angular/core';
import { NavService } from '../header/nav.service';
import { PostModel } from '../shared/post-model';
import { PostService } from '../shared/post.service';

@Component({
  selector: 'app-admin',
  templateUrl: './admin.component.html',
  styleUrls: ['./admin.component.scss']
})
export class AdminComponent implements OnInit {

  constructor(private navService:NavService, private
postService:PostService) { }
 posts: PostModel[];
  ngOnInit(): void {
    this.navService.isAdmin=true;
    this.postService.getAllPosts().subscribe(
      data=>{
        this.posts=data;
      }
    )

  }

}
```

- create-post.component.ts

```
import { Component, OnInit } from '@angular/core';
import { FormGroup, Validators, FormControl } from '@angular/forms';
import { SubredditModel } from 'src/app/subreddit/subreddit-response';
import { Router } from '@angular/router';
import { PostService } from 'src/app/shared/post.service';
import { SubredditService } from 'src/app/subreddit/subreddit.service';
import { throwError } from 'rxjs';
import { CreatePostPayload } from './create-post.payload';

@Component({
```

```
  selector: 'app-create-post',
  templateUrl: './create-post.component.html',
  styleUrls: ['./create-post.component.css']
})
export class CreatePostComponent implements OnInit {

  createPostForm: FormGroup;
  postPayload: CreatePostPayload;
  subreddits: Array<SubredditModel>;

  constructor(private router: Router, private postService: PostService,
    private subredditService: SubredditService) {
    this.postPayload = {
      postName: '',
      url: '',
      description: '',
      subredditName: ''
    }
  }

  ngOnInit() {
    this.createPostForm = new FormGroup({
      postName: new FormControl('', Validators.required),
      subredditName: new FormControl('', Validators.required),
      url: new FormControl('', Validators.required),
      description: new FormControl('', Validators.required),
    });
    this.subredditService.getAllSubreddits().subscribe((data) => {
      this.subreddits = data;
    }, error => {
      throwError(error);
    });
  }

  createPost() {
    this.postPayload.postName                                          =
this.createPostForm.get('postName').value;
    this.postPayload.subredditName                                     =
this.createPostForm.get('subredditName').value;
    this.postPayload.url = this.createPostForm.get('url').value;
    this.postPayload.description                                       =
this.createPostForm.get('description').value;

    this.postService.createPost(this.postPayload).subscribe((data) => {
      this.router.navigateByUrl('/');
```

```
    }, error => {
      throwError(error);
    })
  }

  discardPost() {
    this.router.navigateByUrl('/');
  }

}
```

- app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { SignupComponent } from './auth/signup/signup.component';
import { LoginComponent } from './auth/login/login.component';
import { HomeComponent } from './home/home.component';
import { CreatePostComponent } from './post/create-post/create-post.component';
import { CreateSubredditComponent } from './subreddit/create-subreddit/create-subreddit.component';
import { ListSubredditsComponent } from './subreddit/list-subreddits/list-subreddits.component';
import { ViewPostComponent } from './post/view-post/view-post.component';
import { UserProfileComponent } from './auth/user-profile/user-profile.component';
import { AuthGuard } from './auth/auth.guard';
import { AdminComponent } from './admin/admin.component';

const routes: Routes = [
  { path: '', component: HomeComponent },
  { path: 'view-post/:id', component: ViewPostComponent },
  { path: 'user-profile/:name', component: UserProfileComponent, canActivate: [AuthGuard] },
  { path: 'list-subreddits', component: ListSubredditsComponent },
  // {path: 'view-subreddit/:id',component:ListSubredditsComponent},
  { path: 'create-post', component: CreatePostComponent, canActivate: [AuthGuard] },
  { path: 'create-subreddit', component: CreateSubredditComponent, canActivate: [AuthGuard] },
  { path: 'sign-up', component: SignupComponent },
  { path: 'login', component: LoginComponent },
  {path:'admin',component:AdminComponent,canActivate:[AuthGuard]}
];
```

```
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

- app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HeaderComponent } from './header/header.component';
import { SignupComponent } from './auth/signup/signup.component';
import { ReactiveFormsModule } from '@angular/forms';
import { HttpClientModule, HTTP_INTERCEPTORS } from
'@angular/common/http';
import { LoginComponent } from './auth/login/login.component';
import { NgxWebstorageModule } from 'ngx-webstorage';
import { BrowserAnimationsModule } from '@angular/platform-
browser/animations';
import { ToastrModule } from 'ngx-toastr';
import { TokenInterceptor } from './token-interceptor';
import { HomeComponent } from './home/home.component';
import { FontAwesomeModule } from '@fortawesome/angular-fontawesome';
import { PostTileComponent } from './shared/post-tile/post-
tile.component';
import { VoteButtonComponent } from './shared/vote-button/vote-
button.component';
import { SideBarComponent } from './shared/side-bar/side-bar.component';
import { SubredditSideBarComponent } from './shared/subreddit-side-
bar/subreddit-side-bar.component';
import { CreateSubredditComponent } from './subreddit/create-
subreddit/create-subreddit.component';
import { CreatePostComponent } from './post/create-post/create-
post.component';
import { ListSubredditsComponent } from './subreddit/list-
subreddits/list-subreddits.component';
import { EditorModule } from '@tinymce/tinymce-angular';
import { ViewPostComponent } from './post/view-post/view-post.component';
import { NgbModule } from '@ng-bootstrap/ng-bootstrap';
import { UserProfileComponent } from './auth/user-profile/user-
profile.component';
import { AdminComponent } from './admin/admin.component';
import { ListUserComponent } from './list-user/list-user.component';
```

```
@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    SignupComponent,
    LoginComponent,
    HomeComponent,
    PostTileComponent,
    VoteButtonComponent,
    SideBarComponent,
    SubredditSideBarComponent,
    CreateSubredditComponent,
    CreatePostComponent,
    ListSubredditsComponent,
    ViewPostComponent,
    UserProfileComponent,
    AdminComponent,
    ListUserComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    ReactiveFormsModule,
    HttpClientModule,
    NgxWebstorageModule.forRoot(),
    BrowserAnimationsModule,
    ToastrModule.forRoot(),
    FontAwesomeModule,
    EditorModule,
    NgbModule
  ],
  providers: [
    {
      provide: HTTP_INTERCEPTORS,
      useClass: TokenInterceptor,
      multi: true
    }
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```