**Міністерство освіти і науки України**

**Національний університет «Запорізька Політехніка»**

Кафедра програмних засобів

**Самостійна робота № 2**

з дисципліни «Soft skills, групова динаміка та комунікації» на тему:

«Командна розробка програмного забезпечення»

Виконав:

студент групи КНТ-113сп Іван ЩЕДРОВСЬКИЙ

Прийняв:

доцент Валерій ЛЬОВКІН

2024

# 1 КОМАНДНА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

* 1. **Тема та мета роботи**

Темою роботи - «Pomodoro таймер з використанням Vue»

Метою є навчитися розробляти програмне забезпечення, працюючи в команді. А також навчитися оформлювати програмну документацію.

* 1. **Завдання роботи**
     1. Ознайомитися з основними теоретичними відомостями за темою роботи, використовуючи дані методичні вказівки, а також рекомендовану літературу.
     2. Сформувати команду з двох студентів та отримати у викладача індивідуальне завдання.
     3. Завести акаунт на сервісі GitHub та створити команду для  роботи в Slack.
     4. Узгодити розподіл зобов’язань для виконання завдання,  використовуючи Slack.
     5. Створити репозиторій для роботи над проектом та дозволити доступ до проекту обом користувачам.
     6. Завантажити у віддалений репозиторій існуючі файли проекту.
     7. Створити дві гілки проекту: для презентації ревізій та для  відлагодження проекту.
     8. Налаштувати доступ до Git-репозиторію в інтегрованому  середовищі розробки Eclipse.
     9. Виконати реалізацію проекту в Eclipse та зазначити, яка  частина коду ким була розроблена.
     10. Визначити додаткову функцію та реалізувати її, працюю чи над нею одночасно вдвох.
     11. Завершити роботу над проектом.
     12. Відповідно до діючих стандартів індивідуально оформити  програмний документ, узгоджений з викладачем, на розроблене програмне забезпечення.
     13. Оформити звіт з роботи.
     14. Відповісти на контрольні запитання.

## Текст програми

Тест програми наведений в Додатку А

## Опис розподілу відповідальності в команді

Для виконання цієї самостійної роботи я взяв готовий застосунок, який потрібно було доробити

Оскільки я роблю це завдання один, я створив 2 аккаунти та намагався максимально симулювати командну роботу.

Перший аккаунт відповідав за базове створення застосунку

Другий аккаунт – за тестування застосунок та пошук багів

## Копії екранних форм з результатами виконання завдань

Для виконання цього завдання було створено slack та github, це показано на рисунках 1 та 2 відповідно

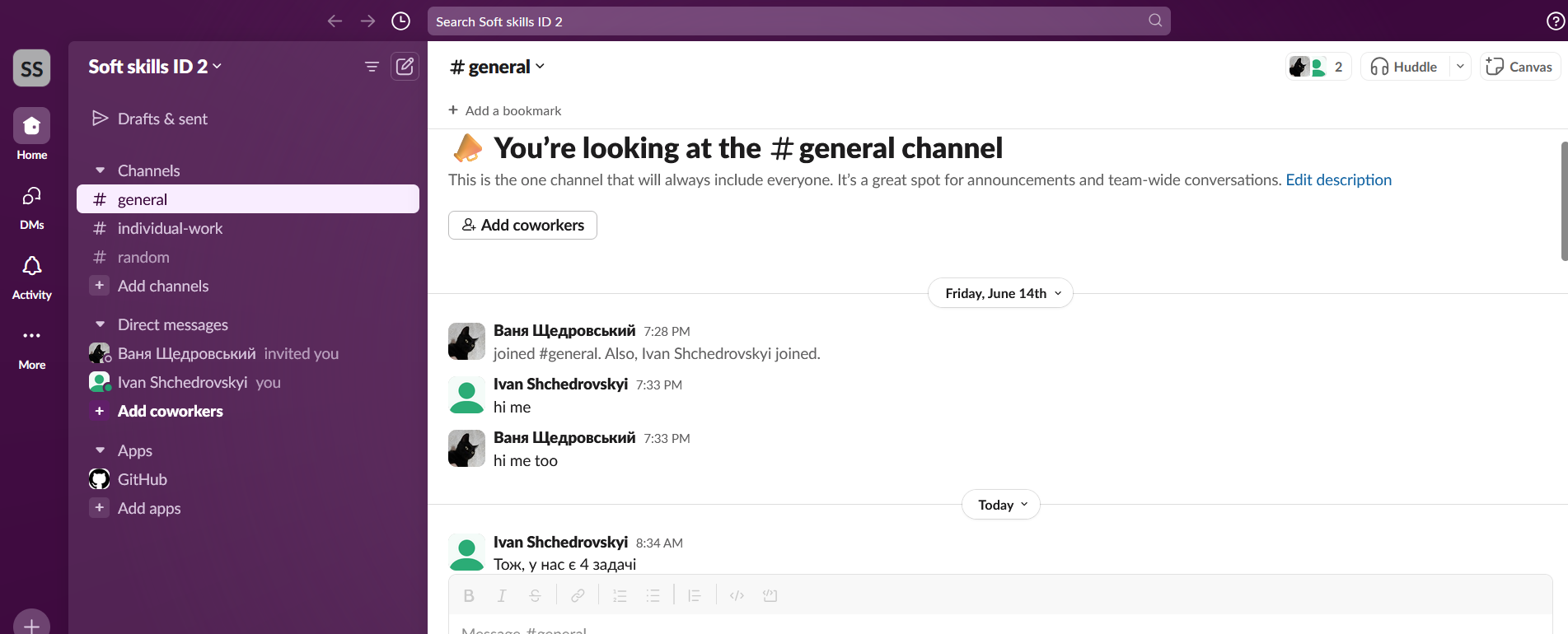


Рисунок 1 – Створений slack

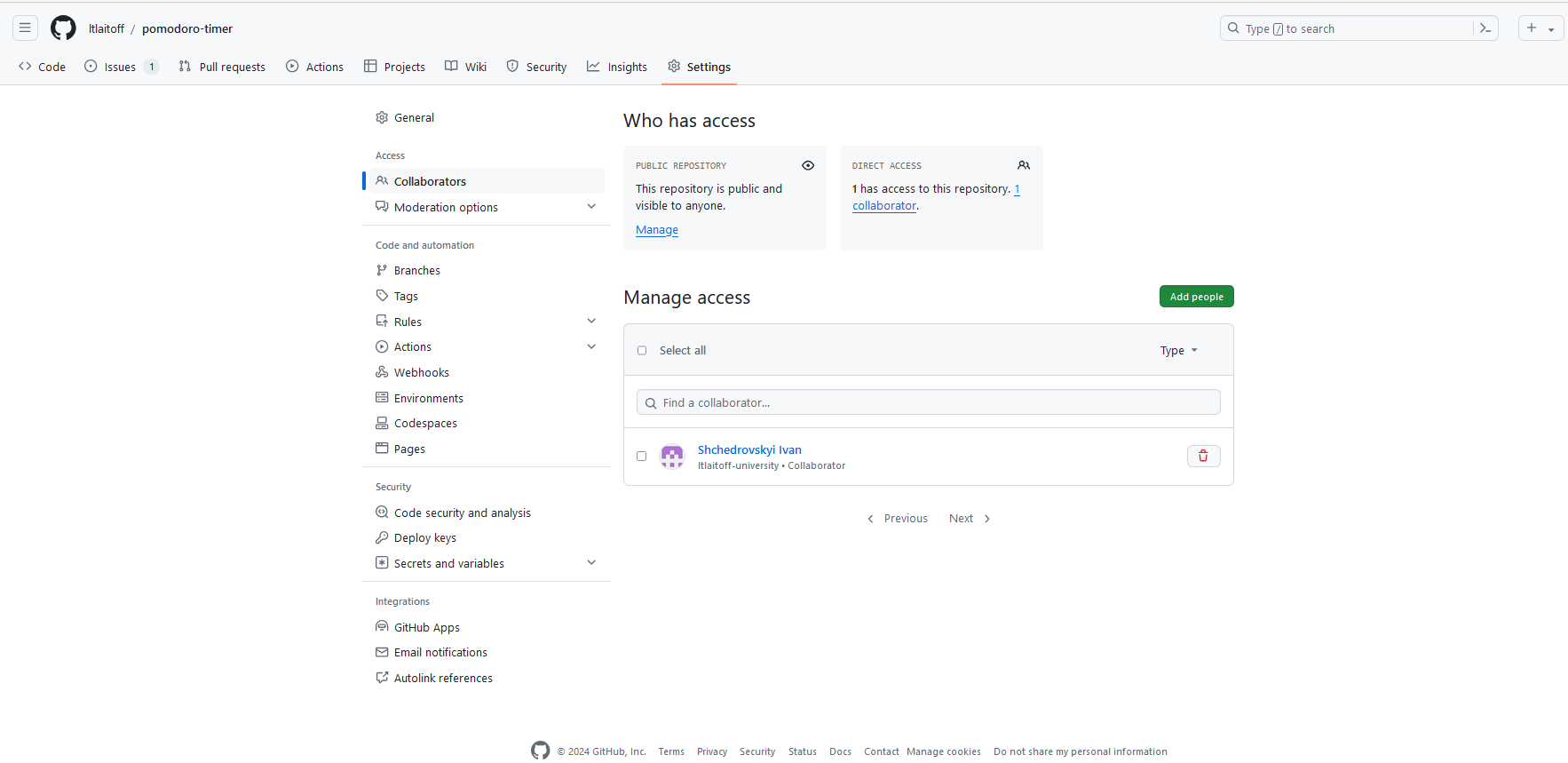


Рисунок 2 – Створений та підключений до обох акаунтів github репозиторій

Далі було створено 4 задачі, два маленьких баги та 2 фічі. Це показано на рисунку 3

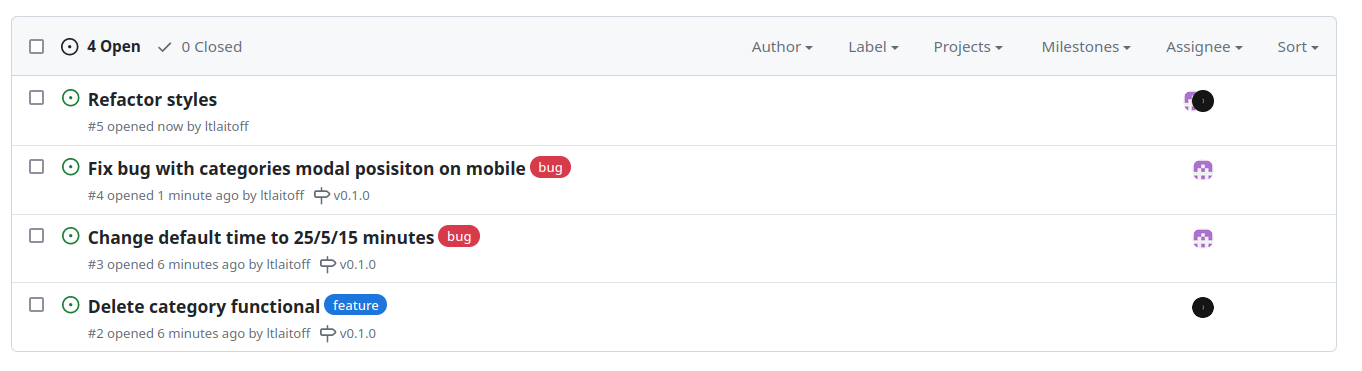


Рисунок 3 – Створення 4-х задач

А також було написано повідомлення в slack щодо цих задач. Це показано на рисунку 4.

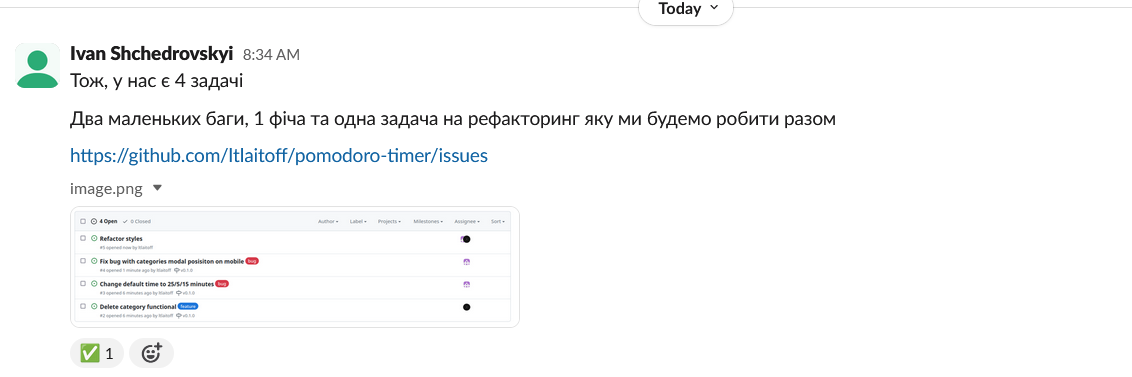


Рисунок 4 – Повідомлення в slack щодо задач

Далі було виконано перше завдання з категоріями через перший аккаунт. Це показано на рисунках 5 та 6.

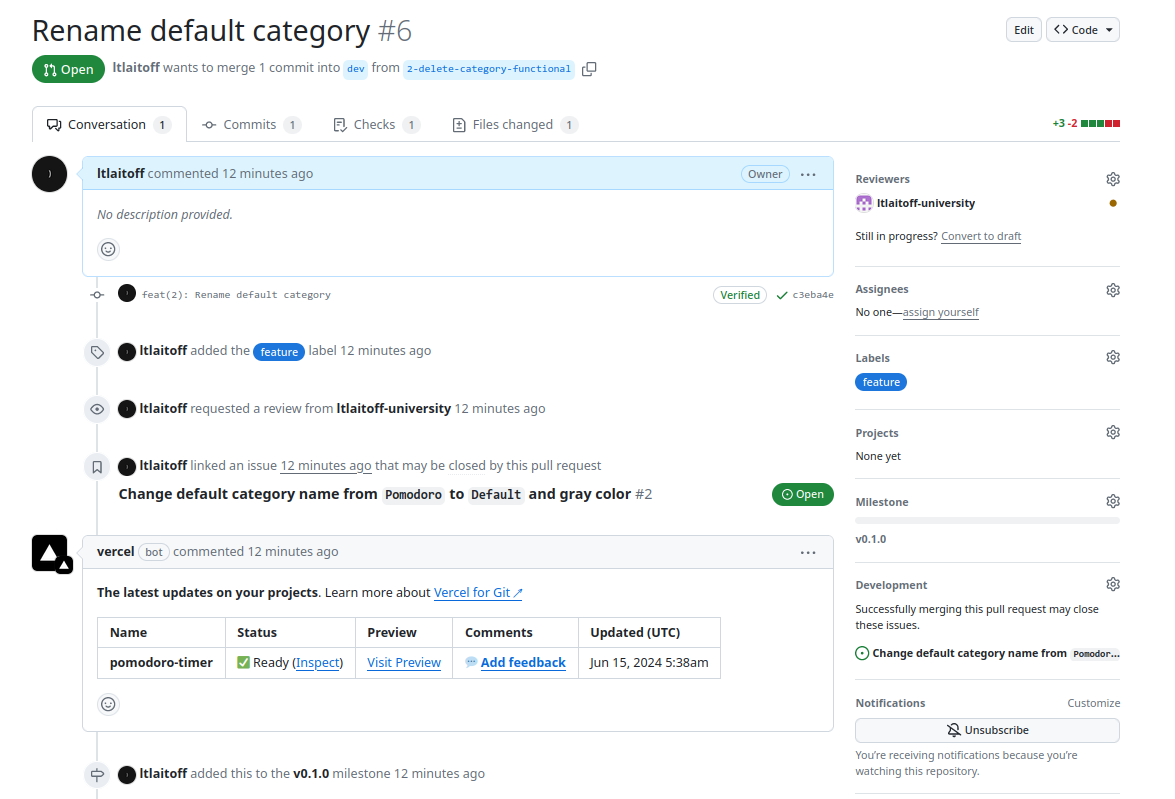


Рисунок 5 – Створення Pull request для виправлення категорій

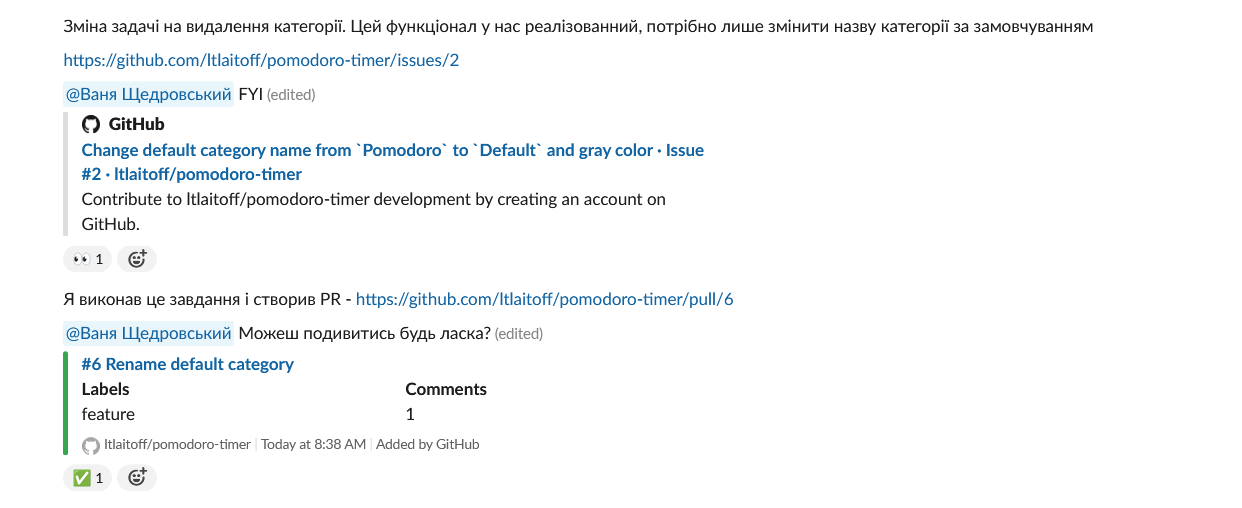


Рисунок 6 – Повідомлення в slack щодо виконаного завдання та запит review

Для git було використано CLI, оскільки вона зручніше. Це показано на рисунку 7



Рисунок 7 – Створення комітів для виправлення категорій

Також було виконано завдання на виправлення часу, але вже з другого аккаунту. Це показано на рисунках 8 та 9.

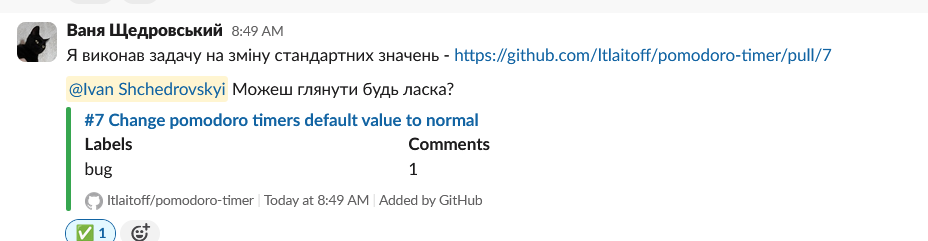


Рисунок 8 – Повідомлення в slack щодо виконаного завдання

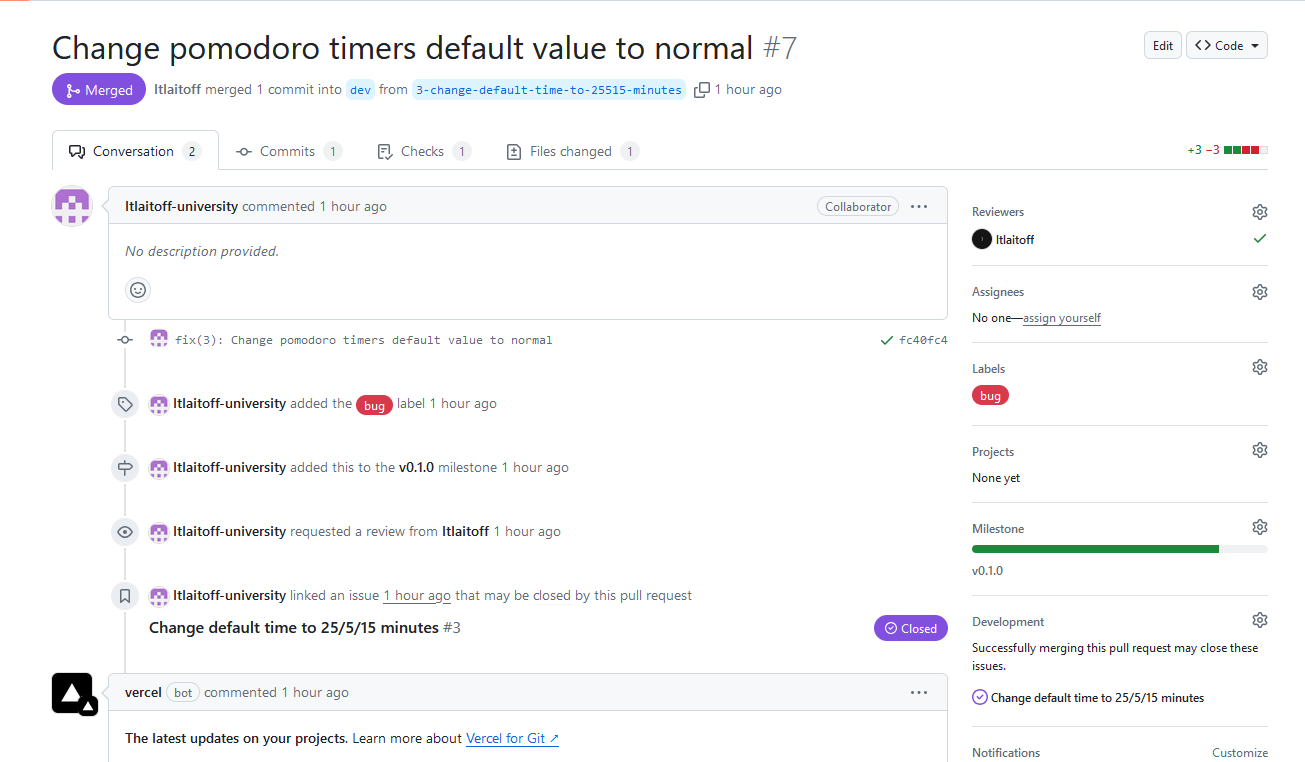


Рисунок 9 – Pull request на виправлення багу

Тепер перейдемо до найбільш цікавої задачі – задачі на встановлення i18n, яку виконували одночасно обидва аккаунти

Це відбувалось таким чином:

* другий аккаунт встановлює базові залежності;
* перший аккаунт встановлю базові перевірки;
* перший аккаунт робить базові тексти;
* другий аккаунт завершує робити тексти;

Виконання цього завдання показано на рисунках 10, 11 та 12

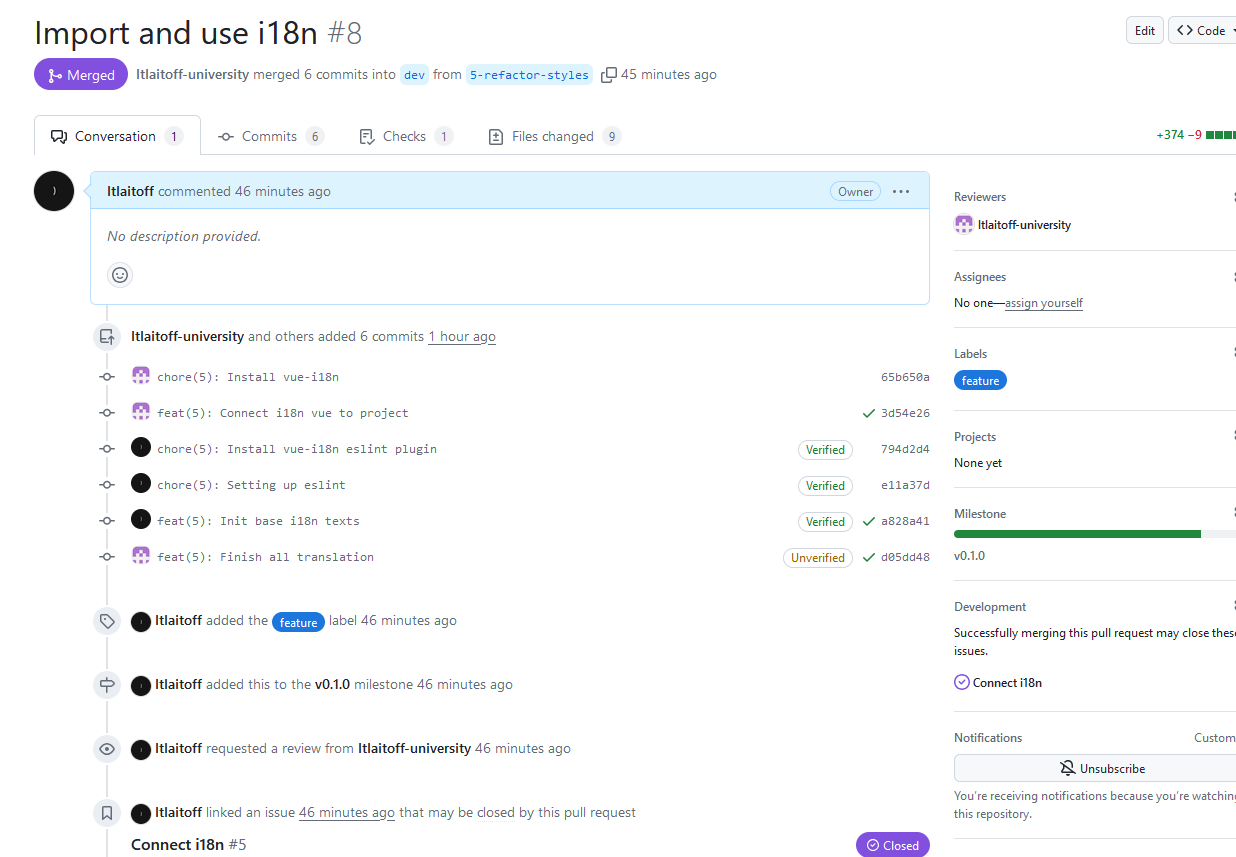


Рисунок 10 – Створення Pull request на виконання завдання декількома акаунтами

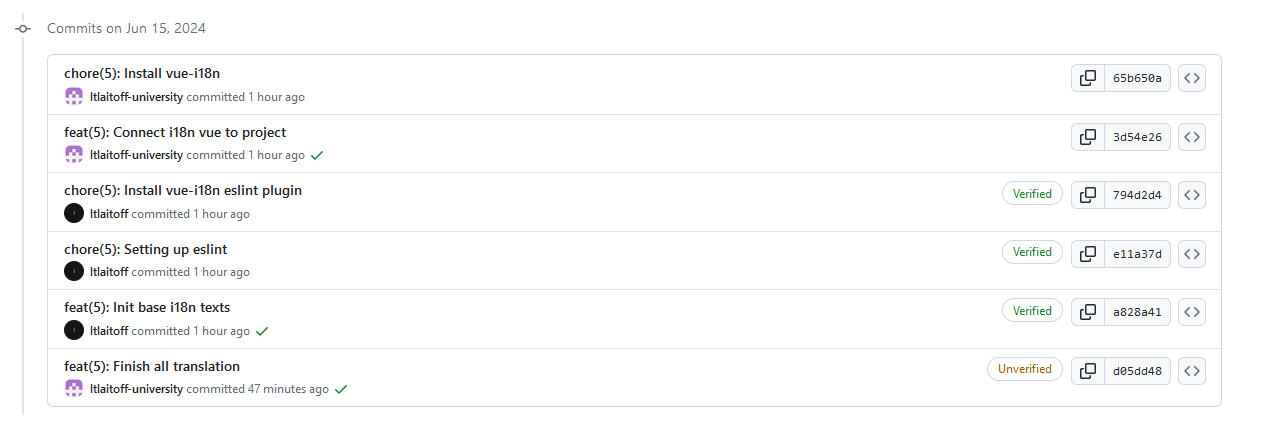


Рисунок 11 – Історія комітів виконання завдання декількома акаунтами

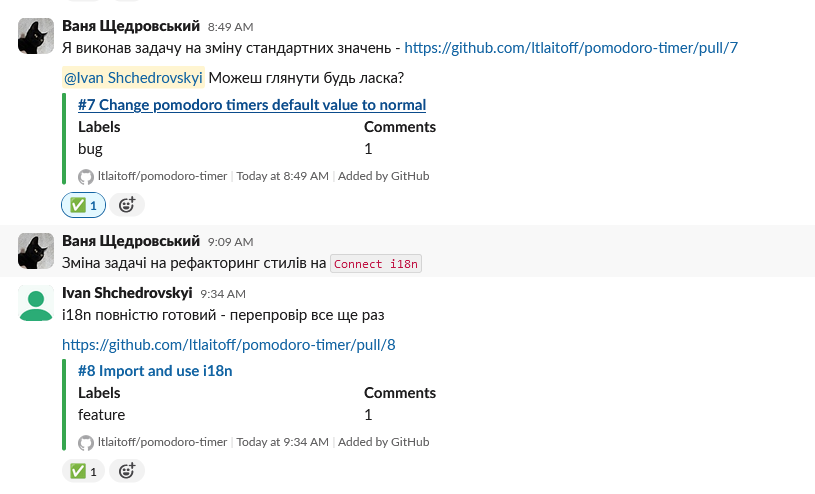


Рисунок 12 – Повідомлення до slack щодо виконання завдання декількома акаунтами

Після merge цієї задачі ми готові робити release

Для цього робить окрема гілка staging/{version}, або ж просто staging. Створюється PR в гілку release, а також на останній комміт додається тег з версією.

Це показано на рисунках 13 та 14

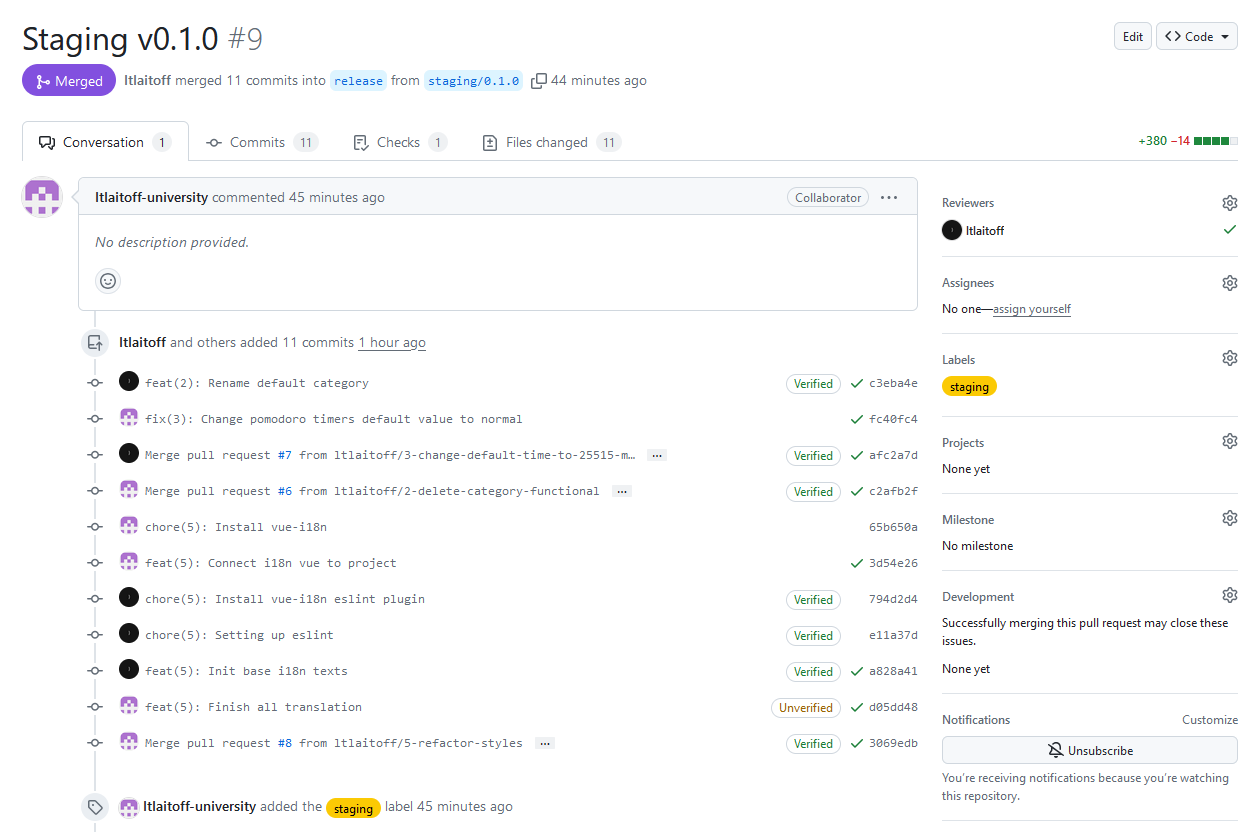


Рисунок 13 - Staging

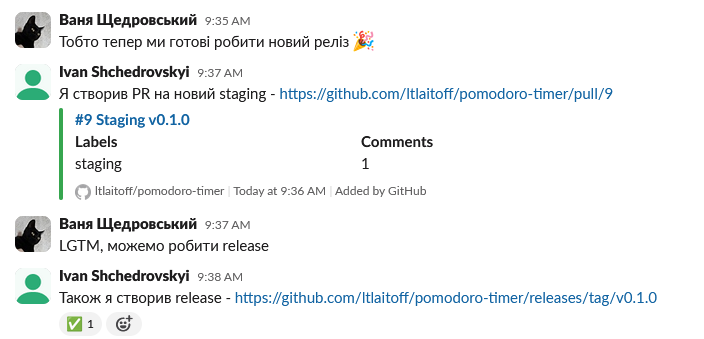


Рисунок 14 – Повідомлення в slack про staging

І після merge цього pull request був зроблений новий release на тег, це показано на рисунку 15.

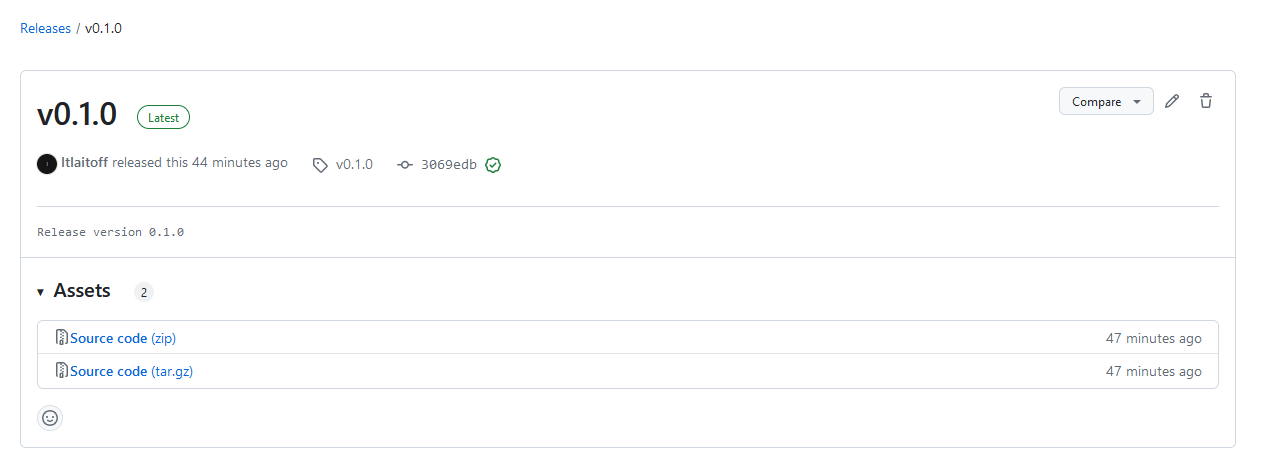


Рисунок 15 – Створення нового release

Вигляд git граф через VS Code та через Webstorm показаний на рисунках 16 та 17.

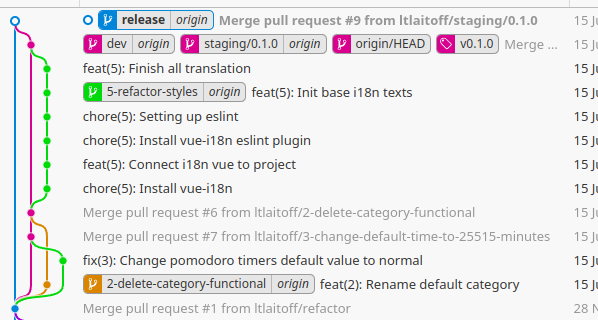


Рисунок 16 – Вигляд git graph через розширення в VS Code



Рисунок 17 – Вигляд git graph через webstorm

## Оформлений програмний документ

Я оформлював технічне завдання

Оформлений програмний документ подається в вигляді окремого файлу до звіту

## Висновки

Я навчився розробляти програмне забезпечення, працюючи в команді. А також навчився оформлювати програмну документацію.

Контрольні запитання

Які основні документи оформлюються при розробленні програмного забезпечення?

* технічне завдання;
* специфікація;
* опис програми;
* текст програми;
* керівництво програміста;
* керівництво системного програміста;
* керівництво оператора.

Які документи на Ваш погляд не є обов’язковими?

* керівництво системного програміста;
* керівництво оператора;
* текст програми;

З написання якого документу починається процес розроблення програмного забезпечення?

Технічне завдання

Які розділи містить технічне завдання?

* вступ;
* підстави для розробки;
* призначення розробки;
* вимоги до програми чи програмному виробу;
* вимоги до програмної документації;
* техніко-економічні показники;
* стадії та етапи розробки;
* порядок контролю та приймання.

З яких розділів складається специфікація?

* документація;
* комплекси;
* компоненти.

**Додаток A**

src\App.vue

<script setup lang="ts">

import tailwindColors from 'tailwindcss/colors'

import LeftPanel from '@components/LeftPanel/LeftPanel.vue'

import HomePage from '@pages/HomePage.vue'

import { useUserSettingsStore } from '@store/userSettingsStore'

const userSettings = useUserSettingsStore()

</script>

<template>

<main class="main font-['Roboto'] w-full h-full min-h-screen pb-[15vh]">

<LeftPanel />

<div class="min-h-full flex items-center justify-center">

<HomePage />

</div>

</main>

</template>

<style scoped>

.main {

background: v-bind(

'tailwindColors[userSettings.colors[userSettings.settings.selectedMode]]["50"]'

);

}

</style>

src\components\CategorySelect.vue

<script setup lang="ts">

import { ref } from 'vue'

import tailwindColors from 'tailwindcss/colors'

import { useCategoryStore } from '@store/categoriesStore'

import { Category } from '@types'

const props = defineProps<{

modelValue: Category

}>()

const emit = defineEmits<{

(e: 'update:modelValue', category: Category): void

}>()

const categoryStore = useCategoryStore()

const opened = ref(false)

function onChange(category: Category) {

emit('update:modelValue', category)

// TODO: Add close on escape

opened.value = false

}

function toggleOpened() {

opened.value = !opened.value

}

function hideSelect() {

opened.value = false

}

</script>

<template>

<div

class="flex justify-center relative"

v-close-modal="hideSelect"

>

<button

class="item rounded-full inline-flex gap-x-2 justify-center items-center px-4 py-2 hover:bg-blue-100 hover:cursor-pointer transition-all duration-200"

:style="{

'--category-select-cicle-bg':

tailwindColors[props.modelValue.color]['400'],

'--category-select-bg-hover':

tailwindColors[props.modelValue.color]['100']

}"

@click="toggleOpened"

>

<div class="circle w-3 h-3 rounded-full item-colors"></div>

<div class="">{{ props.modelValue.name }}</div>

</button>

<div

v-if="opened"

class="absolute max-h-[200px] overflow-scroll top-10 mt-2 flex flex-col rounded-xl overflow-hidden border border-black z-10"

>

<button

v-for="item in categoryStore.categories"

:key="item.\_id"

@click="() => onChange(item)"

class="item flex items-center gap-x-2 px-4 py-2 cursor-pointer transition-all duration-100"

:style="{

'--category-select-cicle-bg': tailwindColors[item.color]['400'],

'--category-select-color': tailwindColors[item.color]['100'],

'--category-select-bg-hover': tailwindColors[item.color]['200']

}"

>

{{ item.name }}

</button>

</div>

</div>

</template>

<style scoped>

.circle {

background: var(--category-select-cicle-bg);

}

.item {

background: var(--category-select-color);

}

.item:hover {

background: var(--category-select-bg-hover);

}

</style>

src\components\ColorPicker.vue

<script setup lang="ts">

import traiwindColors from 'tailwindcss/colors'

import { tailwindColorsIgnore } from '@constants/tailwindColorsIgnore'

import { Colors } from '@types'

const props = withDefaults(

defineProps<{

value: Colors

size?: 'sm' | 'md'

}>(),

{

size: 'md'

}

)

const emits = defineEmits<{

(event: 'change', value: Colors): void

}>()

const colorsForOutput = Object.entries(traiwindColors).filter(([key]) => {

return !tailwindColorsIgnore.includes(key)

// eslint-disable-next-line

}) as [Colors, any][]

function onClick(value: Colors) {

emits('change', value)

}

</script>

<template>

<div class="flex flex-wrap gap-x-2 gap-y-2">

<template

v-for="item in colorsForOutput"

:key="item.toString()"

>

<button

type="button"

class="rounded-full"

:class="{

'w-6 h-6': props.size !== 'sm',

'w-5 h-5': props.size === 'sm'

}"

:style="{

background: item[1]['400'],

border: props.value === item[0] ? '2px solid black' : ''

}"

@click="onClick(item[0])"

></button>

</template>

</div>

</template>

<style scoped></style>

src\components\ControlButtons.vue

<script setup lang="ts">

import tailwindColors from 'tailwindcss/colors'

import FastForwardIcon from '@assets/icons/fastForward.svg'

import PauseIcon from '@assets/icons/pause.svg'

import PlayIcon from '@assets/icons/play.svg'

import StopIcon from '@assets/icons/stop.svg'

import { Colors } from '@types'

const props = defineProps<{

isRunning: boolean

color: Colors

}>()

const emits = defineEmits<{

(e: 'stop'): void

(e: 'fastForward'): void

(e: 'playOrPause'): void

}>()

</script>

<template>

<div class="flex justify-center items-center gap-x-2">

<div class="">

<button

@click="emits('stop')"

class="secondary-button inline text-lg px-4 py-4 rounded-2xl hover:shadow transition-all duration-200"

>

<StopIcon class="w-5 h-5" />

</button>

</div>

<div class="">

<button

@click="emits('playOrPause')"

class="primary-button inline text-lg px-6 py-4 rounded-2xl hover:shadow transition-all duration-200"

>

<PlayIcon

v-if="!props.isRunning"

class="w-7 h-7"

/>

<PauseIcon

v-else

class="w-7 h-7"

/>

</button>

</div>

<div class="">

<button

@click="emits('fastForward')"

class="secondary-button inline text-lg px-4 py-4 rounded-2xl hover:shadow transition-all duration-200"

>

<FastForwardIcon class="w-5 h-5" />

</button>

</div>

</div>

</template>

<style scoped>

.primary-button {

background: v-bind("tailwindColors[props.color]?.['400']");

color: v-bind("tailwindColors[props.color]?.['950']");

}

.primary-button:hover {

background: v-bind("tailwindColors[props.color]?.['500']");

}

.secondary-button {

background: v-bind("tailwindColors[props.color]?.['200']");

color: v-bind("tailwindColors[props.color]?.['900']");

}

.secondary-button:hover {

background: v-bind("tailwindColors[props.color]?.['300']");

}

</style>

src\components\ModeShild.vue

<script setup lang="ts">

import tailwindColors from 'tailwindcss/colors'

import { modeText } from '@constants'

import { useUserSettingsStore } from '@store/userSettingsStore'

import { Mode } from '@types'

const props = defineProps<{

mode: Mode

}>()

const userSettings = useUserSettingsStore()

</script>

<template>

<div>

<div

class="inline-block px-4 py-1 rounded-2xl"

:style="{

background: tailwindColors[userSettings.colors[props.mode]]['100'],

color: tailwindColors[userSettings.colors[props.mode]]['950']

}"

>

{{ modeText[props.mode] }}

</div>

</div>

</template>

src\components\NumberInput.vue

<script setup lang="ts">

import { computed, withDefaults } from 'vue'

import MinusIcon from '@assets/icons/minus.svg'

import PlusIcon from '@assets/icons/plus.svg'

const props = withDefaults(

defineProps<{

modelValue: number

min?: number

max?: number

}>(),

{

min: 0,

max: 1000

}

)

const emits = defineEmits<{

(event: 'update:modelValue', value: number): void

}>()

const inputValue = computed({

get() {

return props.modelValue

},

set(value) {

emits('update:modelValue', validateValue(value))

}

})

const increment = () => {

inputValue.value = props.modelValue + 1

}

const decrement = () => {

inputValue.value = props.modelValue - 1

}

const validateValue = (value: number) => {

if (props.min != undefined && value < props.min) return props.min

if (props.max != undefined && value > props.max) return props.max

return value

}

</script>

<template>

<div class="flex gap-x-2">

<button

class="p-2 bg-slate-200 rounded-xl text-slate-700 hover:bg-slate-300"

@click="decrement"

>

<MinusIcon class="w-4 h-4" />

</button>

<input

class="border border-slate-400 px-2 py-1 rounded-xl"

type="number"

:min="props.min"

:max="props.max"

v-model="inputValue"

/>

<button

class="p-2 bg-slate-200 rounded-xl text-slate-700 hover:bg-slate-300"

@click="increment"

>

<PlusIcon class="w-4 h-4" />

</button>

</div>

</template>

<style scoped></style>

src\components\SelectMode.vue

<script setup lang="ts">

import { computed, ref } from 'vue'

import tailwindColors from 'tailwindcss/colors'

import BedIcon from '@assets/icons/bed.svg'

import BrainIcon from '@assets/icons/brain.svg'

import CupIcon from '@assets/icons/cup.svg'

import { Colors, Mode, ModeItem } from '@types'

const props = defineProps<{

modelValue: Mode

colors: Record<Mode, Colors>

}>()

const emit = defineEmits<{

(e: 'update:modelValue', value: Mode): void

}>()

const selectedColor = computed({

get() {

const findedItem = colorsData.find(item => item.id === props.modelValue)

return findedItem || colorsData[0]

},

set(value) {

emit('update:modelValue', value.id)

}

})

const colorsData: ModeItem[] = [

{

id: Mode.pomodoro,

title: 'Focus',

icon: BrainIcon

},

{

id: Mode.short,

title: 'Short break',

icon: CupIcon

},

{

id: Mode.long,

title: 'Long break',

icon: BedIcon

}

]

function hideSelect() {

selectOpen.value = false

}

function toggleSelect() {

selectOpen.value = !selectOpen.value

}

const selectOpen = ref(false)

function changeSelectedColor(data: ModeItem) {

selectedColor.value = data

selectOpen.value = false

}

</script>

<template>

<div

class="relative whitespace-nowrap"

v-close-modal="hideSelect"

>

<button

class="item-colors rounded-full border flex items-center gap-x-2 px-4 py-2 item-colors-selected cursor-pointer transition-all duration-100"

:style="{

'--select-mode-bg':

tailwindColors[props.colors[selectedColor.id]]['100'],

'--select-mode-border-color':

tailwindColors[props.colors[selectedColor.id]]['950'],

'--select-mode-color':

tailwindColors[props.colors[selectedColor.id]]['950'],

'--select-mode-bg-hover':

tailwindColors[props.colors[selectedColor.id]]['200'],

'--select-mode-selected-border-color':

tailwindColors[props.colors[selectedColor.id]]['950']

}"

@click="toggleSelect"

>

<component

:is="selectedColor.icon"

class="w-6 h-6"

/>

{{ selectedColor.title }}

</button>

<div

v-if="selectOpen"

class="absolute top-10 mt-2 flex flex-col rounded-xl overflow-hidden border border-black z-10"

>

<button

v-for="item in colorsData"

:key="item.title"

class="item-colors flex items-center gap-x-2 px-4 py-2 cursor-pointer transition-all duration-100"

@click="changeSelectedColor(item)"

:style="{

'--select-mode-bg': tailwindColors[props.colors[item.id]]['100'],

'--select-mode-border-color':

tailwindColors[props.colors[item.id]]['950'],

'--select-mode-color': tailwindColors[props.colors[item.id]]['950'],

'--select-mode-bg-hover':

tailwindColors[props.colors[item.id]]['200'],

'--select-mode-selected-border-color':

tailwindColors[props.colors[item.id]]['950']

}"

>

<component

:is="item.icon"

class="w-6 h-6"

></component>

{{ item.title }}

</button>

</div>

</div>

</template>

<style scoped>

.item-colors {

background: var(--select-mode-bg);

border-color: var(--select-mode-border-color);

color: var(--select-mode-color);

}

.item-colors:hover {

background: var(--select-mode-bg-hover);

}

.item-colors-selected {

border-color: var(--select-mode-selected-border-color);

}

</style>

src\components\TimerDigit.vue

<script setup lang="ts">

import { computed } from 'vue'

const props = defineProps<{

digit: number

}>()

const stringDigit = computed(() => {

const digit = props.digit

if (digit >= 10) return String(digit)

return '0' + String(digit)

})

</script>

<template>

<span>

{{ stringDigit }}

</span>

</template>

src\components\TimerTime.vue

<script setup lang="ts">

import TimerDigit from './TimerDigit.vue'

const props = defineProps<{

hours: number

minutes: number

seconds: number

}>()

</script>

<template>

<div

class="text-center mt-6 text-[8rem] max-md:text-[4rem] leading-[0.8] flex items-center"

>

<TimerDigit

v-if="props.hours > 0"

:digit="props.hours"

/>

<span v-if="props.hours > 0">:</span>

<TimerDigit :digit="props.minutes" />

<span>:</span>

<TimerDigit :digit="props.seconds" />

</div>

</template>

src\components\LeftPanel\CategoryForm.vue

<script setup lang="ts">

import { computed, ref } from 'vue'

import ColorPicker from '@components/ColorPicker.vue'

import { Colors } from '@types'

import { AddNewCategory } from '@types'

const name = ref('')

const color = ref<Colors>('fuchsia')

const emits = defineEmits<{

(event: 'close'): void

(event: 'submit', payload: AddNewCategory): void

}>()

const touched = ref(false)

function submit(e: Event) {

e.preventDefault()

if (name.value.length === 0) {

onTouch()

return

}

emits('submit', {

name: name.value,

color: color.value

})

}

const allowShowError = computed(() => {

return touched.value && name.value.length === 0

})

function onTouch() {

touched.value = true

}

function changeColor(newColor: Colors) {

color.value = newColor

}

</script>

<template>

<form

@submit="submit"

class="absolute max-w-[300px] shadow rounded-lg px-4 py-4 flex flex-col gap-y-6 border border-black"

>

<label class="flex gap-x-2 items-center">

<div class="text-slate-800">{{ $t('default.name') }}:</div>

<input

class="w-full border border-slate-400 px-2 py-1 rounded-md"

:class="{

'border-red-400': allowShowError

}"

type="text"

v-model="name"

@input="onTouch"

/>

</label>

<label class="flex flex-col gap-y-2">

<ColorPicker

size="sm"

:value="color"

@change="changeColor"

/>

</label>

<div

v-if="allowShowError"

class="text-red-600 text-center"

>

{{ $t('validation.not-found') }}

</div>

<button

class="w-full bg-green-400 py-2 rounded-md text-white hover:bg-green-500"

>

{{ $t('default.add') }}

</button>

</form>

</template>

src\components\LeftPanel\CategoryPanel.vue

<script setup lang="ts">

import { ref } from 'vue'

import tailwindColors from 'tailwindcss/colors'

import TrashIcon from '@assets/icons/trash.svg'

import { useCategoryStore } from '@store/categoriesStore'

import { AddNewCategory } from '@types'

import CategoryForm from './CategoryForm.vue'

const modalOpened = ref(false)

const categoryStore = useCategoryStore()

function closeModal() {

modalOpened.value = false

}

function submitForm(payload: AddNewCategory) {

categoryStore.add(payload)

modalOpened.value = false

}

</script>

<template>

<div

class="pl-6 pr-6 py-6 w-[450px] max-md:pb-6 max-md:w-full h-full flex flex-col gap-y-2 overflow-scroll"

>

<div class="text-center text-2xl">{{ $t('category.title') }}</div>

<div

class="relative"

v-close-modal="closeModal"

>

<button

class="border px-4 py-2 bg-white border-slate-400 rounded-xl text-slate-700 hover:bg-slate-100 transition-all duration-200"

@click="modalOpened = !modalOpened"

>

{{ $t('category.add') }}

</button>

<CategoryForm

class="top-12 z-50 bg-white"

@close="closeModal"

@submit="submitForm"

v-if="modalOpened"

/>

</div>

<div class="flex flex-col gap-y-2">

<div

class="flex justify-between transition-all duration-200"

v-for="category of categoryStore.categories"

:key="category.\_id"

>

<div class="flex gap-x-2 items-center">

<div

class="w-3 h-3 rounded-full bg-[--category-panel-bg]"

:style="{

'--category-panel-bg': tailwindColors[category.color]['400']

}"

></div>

<div class="text-lg">{{ category.name }}</div>

</div>

<button

v-if="category.\_id !== '0'"

class="group flex items-center justify-center px-2 py-2 rounded-xl hover:bg-slate-100 transition-all duration-200"

@click="categoryStore.delete(category)"

>

<TrashIcon

class="h-7 w-7 text-slate-100 group-hover:text-red-500 transition-all duration-200"

/>

</button>

</div>

</div>

</div>

</template>

<style scoped></style>

src\components\LeftPanel\LeftPanel.vue

<script setup lang="ts">

import { reactive, ref } from 'vue'

import tailwindColors from 'tailwindcss/colors'

import { useUserSettingsStore } from '@store/userSettingsStore'

import { LeftPanelModes } from '@types'

import CategoryPanel from './CategoryPanel.vue'

import PanelButtons from './PanelButtons.vue'

import SettingsPanel from './SettingsPanel.vue'

import StatisticPanel from './StatisticPanel.vue'

const userSettings = useUserSettingsStore()

const LeftPanelSettings = reactive<{

isOpened: boolean

mode: LeftPanelModes

}>({

isOpened: false,

mode: 'statistic'

})

function onClick(mode: LeftPanelModes) {

if (LeftPanelSettings.mode === mode) {

LeftPanelSettings.isOpened = !LeftPanelSettings.isOpened

return

}

LeftPanelSettings.mode = mode

LeftPanelSettings.isOpened = true

}

const burgerMenuOpened = ref(false)

function toggleBurgerMenu() {

burgerMenuOpened.value = !burgerMenuOpened.value

}

</script>

<template>

<div

class="element absolute h-full transition-all duration-700 -translate-x-[450px] ease-in-out flex z-10 max-md:hidden"

:class="LeftPanelSettings.isOpened ? 'translate-x-0' : ''"

:style="{

'--right-panel-color-200':

tailwindColors[userSettings.colors[userSettings.settings.selectedMode]][

'200'

],

'--right-panel-color-300':

tailwindColors[userSettings.colors[userSettings.settings.selectedMode]][

'300'

],

'--right-panel-color-950':

tailwindColors[userSettings.colors[userSettings.settings.selectedMode]][

'950'

]

}"

>

<div

class="w-[450px] z-10 bg-white border-r-[2px] border-[--right-panel-color-300] shadow"

>

<StatisticPanel v-if="LeftPanelSettings.mode === 'statistic'" />

<SettingsPanel v-if="LeftPanelSettings.mode === 'settings'" />

<CategoryPanel v-if="LeftPanelSettings.mode === 'category'" />

</div>

<PanelButtons

:isOpened="LeftPanelSettings.isOpened"

:mode="LeftPanelSettings.mode"

@click="onClick"

:burgerOpened="false"

@burgerToggle="() => {}"

/>

</div>

<div class="max-md:flex hidden">

<div

class="absolute w-full h-full bg-white border-r-[2px] z-20"

v-if="LeftPanelSettings.isOpened"

>

<StatisticPanel v-if="LeftPanelSettings.mode === 'statistic'" />

<SettingsPanel v-if="LeftPanelSettings.mode === 'settings'" />

<CategoryPanel v-if="LeftPanelSettings.mode === 'category'" />

</div>

<button

v-if="LeftPanelSettings.isOpened"

class="absolute top-5 right-5 w-8 h-8 bg-slate-200 rounded-lg z-50 flex items-center justify-center"

:class="burgerMenuOpened ? 'gap-y-[0px]' : 'gap-y-[5px]'"

@click="() => (LeftPanelSettings.isOpened = false)"

>

<div class="flex flex-col gap-y-[3px]">

<div class="bg-black w-[15px] h-[2px] rounded-full rotate-45"></div>

<div

class="bg-black w-[15px] h-[2px] rounded-full -mt-[5px] -rotate-45"

></div>

</div>

</button>

<PanelButtons

:isOpened="LeftPanelSettings.isOpened"

:mode="LeftPanelSettings.mode"

@click="onClick"

:burgerOpened="burgerMenuOpened"

@burgerToggle="toggleBurgerMenu"

/>

</div>

</template>

src\components\LeftPanel\PanelButtons.vue

<script setup lang="ts">

import tailwindColors from 'tailwindcss/colors'

import { LeftPanelModes } from '@types'

import { useUserSettingsStore } from '../../store/userSettingsStore'

const props = defineProps<{

isOpened: boolean

mode: LeftPanelModes

burgerOpened: boolean

}>()

const emits = defineEmits<{

(event: 'click', value: LeftPanelModes): void

(event: 'burgerToggle'): void

}>()

const userSettings = useUserSettingsStore()

const data: {

title: string

key: LeftPanelModes

}[] = [

{

title: 'Settings',

key: 'settings'

},

{

title: 'Categories',

key: 'category'

},

{

title: 'Statistic',

key: 'statistic'

}

]

</script>

<template>

<div class="pt-10 h-full flex flex-col justify-start gap-y-2 max-md:hidden">

<button

v-for="item of data"

:key="item.key"

class="py-3 px-4 rounded-r-xl shadow clip-your-needful-style text-[--right-panel-color-950]"

:class="

props.isOpened === false

? 'bg-[--right-panel-color-200]'

: props.mode === item.key

? 'bg-[--right-panel-color-300] z-20'

: 'bg-[--right-panel-color-200]'

"

@click="emits('click', item.key)"

>

{{ item.title }}

</button>

</div>

<div class="max-md:flex hidden z-10">

<button

class="absolute top-5 right-5 w-8 h-8 bg-[--button-panel-bg] rounded-lg z-50 flex items-center justify-center"

:class="props.burgerOpened ? 'gap-y-[0px]' : 'gap-y-[5px]'"

:style="{

'--button-panel-bg':

tailwindColors[userSettings.colors[userSettings.selectedMode]]['300']

}"

@click="emits('burgerToggle')"

>

<div class="flex flex-col gap-y-[3px]">

<div

class="bg-black w-[15px] h-[2px] rounded-full"

:class="props.burgerOpened ? ' rotate-45' : 'rotate-0'"

></div>

<div

class="bg-black w-[15px] h-[2px] rounded-full"

:class="props.burgerOpened ? 'hidden' : ''"

></div>

<div

class="bg-black w-[15px] h-[2px] rounded-full"

:class="props.burgerOpened ? '-mt-[5px] -rotate-45' : 'rotate-0'"

></div>

</div>

</button>

<div

class="absolute min-w-[100vw] w-full h-full bg-white flex items-center justify-center flex-col"

:class="props.burgerOpened ? 'top-0' : '-top-full'"

>

<button

v-for="item of data"

:key="item.key"

class="py-6 px-4 text-[--right-panel-color-950] hover:bg-slate-200 rounded-md transition-all duration-150 text-2xl w-full"

@click="emits('click', item.key)"

>

{{ item.title }}

</button>

</div>

</div>

</template>

src\components\LeftPanel\SettingsPanel.vue

<script setup lang="ts">

import ColorPicker from '@components/ColorPicker.vue'

import ModeShild from '@components/ModeShild.vue'

import NumberInput from '@components/NumberInput.vue'

import { useUserSettingsStore } from '@store/userSettingsStore'

import { Mode } from '@types'

const userSettings = useUserSettingsStore()

const modes: Mode[] = Object.values(Mode) as Mode[]

</script>

<template>

<div

class="pl-6 pr-6 py-6 w-[450px] max-md:pb-6 max-md:w-full h-full flex flex-col gap-y-2 overflow-y-scroll"

>

<div class="text-center text-2xl">{{ $t('settings.title') }}</div>

<div class="flex flex-col gap-y-10">

<div class="flex flex-col gap-y-10">

<div

class="flex flex-col gap-y-4"

v-for="mode of modes"

:key="mode"

>

<ModeShild :mode="mode" />

<NumberInput

class="ml-1"

v-model="userSettings.settings.times[mode]"

:max="1439"

:min="1"

/>

<ColorPicker

class="ml-1"

:value="userSettings.colors[mode]"

@change="value => userSettings.setColor(mode, value)"

/>

</div>

<div class="">

<div class="px-4 text-lg">Approaches count:</div>

<NumberInput

class="ml-1 mt-1"

v-model="userSettings.settings.approachesCount"

:max="100"

:min="1"

/>

</div>

</div>

</div>

</div>

</template>

<style scoped></style>

src\components\LeftPanel\StatisticPanel.vue

<script setup lang="ts">

import { computed } from 'vue'

import tailwindColors from 'tailwindcss/colors'

import TrashIcon from '@assets/icons/trash.svg'

import ModeShild from '@components/ModeShild.vue'

import { useStatistic } from '@store/statisticStore'

const statisticStore = useStatistic()

const statisticForOutput = computed(() => {

// Move to helpers

return statisticStore.statistic.toSorted((a, b) => {

return Number(b.date) - Number(a.date)

})

})

</script>

<template>

<div

class="pl-6 pr-6 py-6 w-full h-full flex flex-col gap-y-4 overflow-y-scroll"

>

<div class="text-center text-2xl">{{ $t('statistic.title') }}</div>

<div

class="flex w-full gap-x-2"

v-for="item in statisticForOutput"

:key="item.\_id"

>

<div class="w-full flex flex-col gap-y-1">

<div class="flex justify-between gap-x-5 items-center">

<span class="ml-2 font-bold text-lg">

{{ new Date(item.count).toISOString().substring(14, 19) }}

</span>

<span class="">

{{ new Date(item.date).toLocaleString('en-GB') }}

</span>

</div>

<div class="flex justify-between gap-x-4">

<ModeShild :mode="item.mode" />

<div

class="flex gap-x-2 items-center hover:bg-blue-100 hover:cursor-pointer transition-all duration-200"

>

<div

class="w-3 h-3 rounded-full"

:style="{

backgroundColor: tailwindColors[item.category.color]['400']

}"

></div>

<div class="">{{ item.category.name }}</div>

</div>

</div>

</div>

<div class="flex justify-center items-center">

<button

class="group flex items-center justify-center px-2 py-2 rounded-xl hover:bg-slate-100 transition-all duration-200"

@click="statisticStore.delete(item)"

>

<TrashIcon

class="h-7 w-7 text-slate-100 group-hover:text-red-500 transition-all duration-200"

/>

</button>

</div>

</div>

</div>

</template>

<style scoped></style>

src\pages\HomePage.vue

<script setup lang="ts">

import { computed, onMounted, watchEffect } from 'vue'

import { useTimer } from '@/packages/timer'

import CategorySelect from '@components/CategorySelect.vue'

import ControlButtons from '@components/ControlButtons.vue'

import SelectMode from '@components/SelectMode.vue'

import TimerTime from '@components/TimerTime.vue'

import { useStatistic } from '@store/statisticStore'

import { useUserSettingsStore } from '@store/userSettingsStore'

import { Mode, ModeItem } from '@types'

const userSettings = useUserSettingsStore()

const statisticStore = useStatistic()

const selectedMode = computed(() => userSettings.selectedMode)

const TIMER\_POMODORO\_MODE = computed(

() => userSettings.times[Mode.pomodoro] \* 60 \* 1000

)

const TIMER\_SHORT\_BREAK\_MODE = computed(

() => userSettings.times[Mode.short] \* 60 \* 1000

)

const TIMER\_LONG\_BREAK\_MODE = computed(

() => userSettings.times[Mode.long] \* 60 \* 1000

)

const timer = useTimer(Date.now() + TIMER\_POMODORO\_MODE.value, false)

// TODO: Remove / move to mounted

resetTimer()

// TODO: 50/50

userSettings.$subscribe(data => {

if (

userSettings.activeCompletedPomodoro &&

data &&

data.events &&

// @ts-expect-error All good

data.events.key !== undefined &&

// @ts-expect-error All good

data.events.key === 'selectedMode' &&

// @ts-expect-error All good

data.events.newValue === Mode.pomodoro

) {

incrementAndResetApproach()

userSettings.activeCompletedPomodoro = false

}

resetTimer()

})

function onPlayOrPauseClick() {

if (timer.isRunning.value) {

timer.pause()

return

}

timer.start()

}

function onFastForwardClick() {

const currentTimer =

selectedMode.value === Mode.pomodoro

? TIMER\_POMODORO\_MODE

: selectedMode.value === Mode.short

? TIMER\_SHORT\_BREAK\_MODE

: TIMER\_LONG\_BREAK\_MODE

timer.restart(Date.now() + currentTimer.value, false)

incrementAndResetApproach()

}

function incrementAndResetApproach() {

userSettings.currentApproach++

if (userSettings.currentApproach > userSettings.settings.approachesCount) {

userSettings.currentApproach = 1

}

}

onMounted(() => {

watchEffect(async () => {

if (timer.isExpired.value) {

if (selectedMode.value !== Mode.pomodoro) {

if (selectedMode.value === Mode.short) {

statisticStore.add({

mode: selectedMode.value,

count: TIMER\_LONG\_BREAK\_MODE.value,

category: userSettings.settings.selectedCategory

})

} else {

statisticStore.add({

mode: selectedMode.value,

count: TIMER\_SHORT\_BREAK\_MODE.value,

category: userSettings.settings.selectedCategory

})

}

onSelectedModeChange(Mode.pomodoro)

return

}

userSettings.activeCompletedPomodoro = true

statisticStore.add({

mode: selectedMode.value,

count: TIMER\_POMODORO\_MODE.value,

category: userSettings.settings.selectedCategory

})

if (userSettings.currentApproach >= 4) {

onSelectedModeChange(Mode.long)

} else {

onSelectedModeChange(Mode.short)

}

}

})

})

function onSelectedModeChange(id: ModeItem['id']) {

userSettings.setSelectedMode(id)

resetTimer(id)

}

function resetTimer(mode: Mode = selectedMode.value) {

if (mode === Mode.pomodoro) {

timer.restart(Date.now() + TIMER\_POMODORO\_MODE.value, false)

}

if (mode === Mode.short) {

timer.restart(Date.now() + TIMER\_SHORT\_BREAK\_MODE.value, false)

}

if (mode === Mode.long) {

timer.restart(Date.now() + TIMER\_LONG\_BREAK\_MODE.value, false)

}

}

</script>

<template>

<div class="w-[450px] flex flex-col items-center">

<SelectMode

:model-value="userSettings.selectedMode"

@update:model-value="onSelectedModeChange"

:colors="userSettings.colors"

/>

<div class="py-16 flex flex-col justify-center">

<CategorySelect v-model="userSettings.settings.selectedCategory" />

<TimerTime

:hours="timer.hours.value"

:minutes="timer.minutes.value"

:seconds="timer.seconds.value"

/>

<div class="mt-3 text-lg flex justify-center text-">

<span class="text-blue-800">

{{ userSettings.currentApproach }}

</span>

<span class="text-blue-950">/</span>

<span class="text-blue-950">

{{ userSettings.settings.approachesCount }}

</span>

</div>

</div>

<ControlButtons

@stop="resetTimer"

@play-or-pause="onPlayOrPauseClick"

@fast-forward="onFastForwardClick"

:is-running="timer.isRunning.value"

:color="userSettings.colors[userSettings.settings.selectedMode]"

/>

</div>

</template>