МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2022 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

ВЕБЗАСТОСУНОК ПЛАНУВАННЯ ЗАДАЧ ІЗ ПІДТРИМКОЮ ГРУПОВИХ ЗАХОДІВ

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402. 21810208

Виконала студентка 4-го курсу, групи 402
_____ А. К. Гресс
«22» червня 2022 р.

Керівник: канд. техн. наук, доцент (б.в.з.)
_____М. Л. Дворецький
«22» червня 2022 р.

Миколаїв – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____20____р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

Видано студентці групи 402 факультету комп'ютерних наук Гресс Анастасії Костянтинівні

1. Тема кваліфікаційної роботи «Вебзастосунок планування задач із підтримкою групових заходів».

Керівник роботи Дворецький Михайло Леонідович, канд. техн. наук, доцент (б.в.з.)

Затв. наказом Ректора ЧНУ ім. Петра Могили від «07» 12 2021 р. № 318

2. Строк представлення кваліфікаційної роботи студентом «__» _____ 20__ р.

3. Очікуваний результат роботи: веб-застосунок для планування користувацьких задач із можливістю організації групових заходів.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):
- огляд основних стратегій планування задач;
- аналіз сучасних систем тайм-менеджменту;
- вибір інструментальних засобів створення застосунку;
- програмна реалізація веб-системи для планування задач із підтримкою групових заходів.

5. Перелік графічного матеріалу: сторінок – 76 , таблиць – 2 , рисунків – 35, посилань – 29 , презентація.

6. Завдання до спеціальної частини: «Охорона праці при користуванні екранними пристроями».

7. Консультанти:

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Алексєєва А.А., канд. техн. наук, доцент	

Керівник роботи ___канд. техн. наук, доцент (б.в.з.) Дворецький М._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання _Гресс А. К._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «_23_» ___листопада_____ 2022 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: «Веб-застосунок планування задач із підтримкою групових заходів»

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників БКР	01.11.2021	01.11.2021	виконано
2	Отримання завдання на виконання БКР	23.11.2021	23.11.2021	виконано
3	Складання календарного плану роботи на весь період виконання БКР	06.12.2021	10.12.2021	виконано
4	Отримання завдання на переддипломну практику	23.05.2022	23.05.2022	виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до БКР	24.05.2022	02.06.2022	виконано
6	Розробка звіту з переддипломної практики	03.06.2022	04.06.2022	виконано
7	Виконання БКР: аналіз предметної сфери, вибір інструментальних засобів створення застосунку, розробка веб-додатку, тестування	28.02.2022	29.05.2022	виконано
8	Попередній захист БКР на засіданні комісії кафедри	30.05.2022	30.05.2022	виконано
9	Доробка та остаточне оформлення БКР	31.05.2022	11.06.2022	виконано
10	Подання БКР рецензенту	17.06.2022	17.06.2022	виконано
11	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	24.06.2022	24.06.2022	виконано
12	Захист БКР перед екзаменаційною комісією (ЕК)	29.06.2022	29.06.2022	виконано

Розробив студент _____Гресс А. К._____ _____
(прізвище та ініціали) (*підпис*)

Керівник роботи канд. техн. наук, доцент (б.в.з.) Дворецький М. Л _____
(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

«10» грудня 2022 р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи студентки групи 402 ЧНУ ім. Петра Могили

Гресс Анастасії Костянтинівни

Тема: «Вебзастосунок планування задач із підтримкою групових заходів»

Об’єкт роботи – процеси планування задач та тайм-менеджменту.

Предмет роботи – засоби проєктування та розробки веб-застосунків для ефективного планування й обліку поставлених задач та цілей.

Метою бакалаврської кваліфікаційної роботи є створення передумов для оптимізації процесу персонального тайм-менеджменту шляхом розробки веб-застосунку планування задач із підтримкою групових заходів.

Робота складається з фахового розділу і спеціальної частини з охорони праці. Пояснювальна записка складається зі вступу, чотирьох розділів, висновків та додатків.

У першому розділі розглядається загальний аналіз сфери планування задач та основні стратегії, що застосовуються для тайм-менеджменту.

У другому розділі досліджено сучасні системи тайм-менеджменту, визначено основні вимоги та наведено аналіз існуючих веб-систем.

У третьому розділі розглянуто вимоги до проєктування системи, описано програмну реалізацію за допомогою бібліотеки React JS, а також наведено опис користувацького інтерфейсу веб-застосунка.

В останньому спеціальному розділи було розглянуто нормативну базу охорони праці при користуванні екранними приладами.

В результаті виконаної роботи було розроблено веб-застосунок планування задач із підтримкою групових заходів.

Сторінок – 78 , таблиць – 2 , рисунків – 35 , посилань – 29 , додатків – 3.

Ключові слова: тайм-менеджмент, планування, пріоритет, продуктивність, React JS, SPA.

ABSTRACT

for bachelor's qualification work of a student of 402 group at Petro Mohyla Black Sea National University

Hress Anastasiia Kostiantynivna

Topic: "Web application for task planning with the support of group activities"

The object of the research is the processes of task planning and time management.

The subject of the research is the means of designing and developing web applications for effective planning and accounting of tasks and goals.

The purpose of the bachelor's thesis is to create the conditions for optimizing the process of personal time management by developing a web application for planning tasks with support for group activities.

The research work consists of a professional section and a special section on labor protection. The explanatory note consists of an introduction, four chapters, conclusions and appendices.

The first section discusses a general analysis of the field of task planning and the main strategies used for time management.

The second section examines modern time management systems, defines the basic requirements and analyzes existing web systems.

The third section specifies the requirements for system design, describes the software implementation using the React JS library, and describes the user interface of the web application.

In the last section the normative base of labor protection at use of screen devices was considered.

As the result of the performed work, conclusions …

Pages – 78, tables – 2, figures – 35, links – 29, appendices – 3.

Keywords: time management, planning, priority, productivity, React JS, SPA.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД	– база даних
API	– Application Programming Interface
CSS	– Cascading Style Sheets
DNS	– Domain Name System
DOM	– Document Object Model
HTML	– HyperText Markup Language
HTTP	– HyperText Transfer Protocol
JSX	– JavaScript XML
JVM	– Java Virtual Machine
MPA	– Multi Page Application
NoSQL	– Not only SQL
RTK	– Redux Toolkit
SPA	– Single Page Application
SQL	– Structured Query Language
TS	– TypeScript
UI	– User Interface
UUID	– Universally Unique Identifier
XML	– Extensible Markup Language

Пояснювальна записка

до кваліфікаційної роботи

на тему:

«ВЕБЗАСТОСУНОК ПЛАНУВАННЯ ЗАДАЧ ІЗ ПІДТРИМКОЮ ГРУПОВИХ ЗАХОДІВ»

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402.21810208

Виконала студентка 4-го курсу, групи 402
А. К. Гресс
(підпис, ініціали та прізвище)
«22» червня 2022 р.

Керівник: *канд. техн. наук, доцент (б.в.з.)*
(наук. ступінь, вчене звання)
М. Л. Дворецький
(підпис, ініціали та прізвище)
«22» червня 2022 р.

Миколаїв – 2022

ВСТУП

Життя сучасної людини є досить динамічним і для того, щоб воно і справді вирувало, а не проходило повз із постійними думками про брак часу та можливостей, необхідно вміти планувати та правильно розставляти пріоритети. Якщо замислитись, то кожен з нас має у своєму арсеналі 24 години щодня, але не всі ми вміємо вправно розподіляти цей час. Саме тому тайм-менеджмент є однією із найважливіших навичок, необхідних для щасливого та успішного життя у XXI сторіччі. І найголовніше – людина повинна самостійно підкуватися в області планування своїх особистих та робочих задач, адже ні у школі, ні в університеті даний предмет не вивчається.

Крім того, важливо зазначити, що в останні роки все більше популяризується діджиталізація. Особливо використання цифрових та інноваційних технологій набрало популярності після спалаху епідемії COVID-19, коли велика кількість установ та компаній повинна була перейти у дистанційний режим роботи, і отже, основним інструментом обміну інформації став персональний комп’ютер, а не паперові носії. Саме тому веб-система для результативного планування задач є як ніколи актуальною, зокрема за умови підтримки організації групових заходів, адже такий функціонал дозволяє швидко знайти «вікна» у розкладі людини та згодити їх із своїм вільним часом.

Це обумовило **мету роботи**, яка полягає в створенні передумов для оптимізації процесу персонального тайм-менеджменту шляхом розробки веб-застосунку планування задач із підтримкою групових заходів.

Об’єкт дослідження – процеси планування задач та тайм-менеджменту.

Предмет дослідження – засоби проєктування та розробки веб-застосунків для ефективного планування й обліку поставлених задач та цілей.

Досягнення поставленої мети обумовлює необхідність вирішення наступних **задач**.

1) Розглянути та проаналізувати найпоширеніші методології планування задач.

2) Провести аналіз існуючих веб-систем для тайм-менеджменту та виокремити найбільш ефективні рішення.

3) Зробити вибір інструментальних засобів створення застосунку, опрацювавши особливості розробки веб-додатків, та визначити вимоги для програмного продукту.

4) Спроєктувати структуру та логіку веб-застосунку.

5) Здійснити програмну реалізацію веб-системи для планування задач із підтримкою групових заходів на основі визначених вимог.

1 АНАЛІЗ ПОШИРЕШИХ ТЕХНІК ПЛАНУВАННЯ

1.1 Загальний огляд стратегій планування

Кожного дня ми зіткаємося із певною кількістю необхідних для виконання задач як робочих, так і особистих, однак за результатами дослідження, проведеного у 2021 році лише 18% респондентів користується методикою планування [1]. Інша ж частина або не застосовує жодну із технік тайм-менеджменту, або ж йде по найбільш легкому шляху – складає звичайний список справ, однак зазвичай такий підхід не увінчується успіхом.

Нездатність належним чином керувати своїм часом та вміти ефективно планувати задачі не лише спричиняє зниження рівня працездатності, але й несе за собою такі наслідки, як нервове напруження, стрес, швидка стомлюваність та в деяких випадках навіть вигорання. Нам здається, що у добі занадто мало часу для того, щоб встигнути охопити справи з усіх наших сфер життя, однак насправді ж, якщо ви відчуваєте брак часу, ви просто не вмієте грамотно його розподіляти.

Для того, щоб розібратися, чому так важливо користуватися методиками планування, необхідно виділити переваги, які ми можемо отримати при правильному підході до розподілу свого часу.

По-перше, завдяки раціональному плануванню поліпшується продуктивність та ефективність виконання завдань, що сприяє покращенню робочого іміджу, внаслідок чого підвищується настрій та самооцінка, вигасають стреси.

По-друге, тайм-менеджмент дозволяє почувати себе більш впевнено, адже в будь-якій ситуації у людини буде присутній чіткий план дій, яким можна користуватися задля того, щоб досягати поставлених цілей.

Щодо самого тайм-менеджменту, то його можна розділити на три категорії:

– *персональний*: охоплює аспект самовдосконалення людини, тобто це комплекс практик й технік, які людина використовує для продуктивного планування персонального часу;

– *корпоративний*: формування трудової діяльності, а саме розподіл робочого часу та обов'язків в організації;

– *соціальний*: стосується групи людей і зводиться до регулювання відносин, а не лише полягає у розподілі часу.

У дані роботі всю увагу приділено саме персональному тайм-менеджменту, адже саме він є найбільш прикладним, адже його методики можуть бути застосовані у будь-якому віці та для будь-якої сфери діяльності.

Важливо зазначити те, що планування часу – наука не нова. Не зважаючи на те, що саме в останні роки із стрімким розвитком соціальних мереж суспільство все частіше почало бачити зображення ідеального життя в шаленому ритмі продуктивності з посиланням на те, що запорукою цього успіху є саме правильно поставлений тайм-менеджмент, все ж таки посягання на керування часом виникли ще 2000 років тому. Філософ Сенека визначив базові принципи планування, а саме:

– вести письмовий облік часу;

– оцінювати періоди життя з точки зору ефективності;

– розділяти час на продуктивно проведений або ж змарнований.

Однак все ж таки сучасне поняття «тайм-менеджмент» почало свій розвиток у 70-х роках минулого століття, причому воно поступово еволюціонувало від звичайного поняття ефективного виконання роботи до ідеї гармонійного розподілу часу між всіма сферами життя. І у сьогодення актуальність даного терміну тільки зростає, що зумовлено підвищеним темпом повсякдення, в якому надзвичайно важливо зберігати баланс між роботою та особистим життям.

Крім того, планування задач буде результативним лише умови виконання певних правил. До числа найбільш визнаних рекомендацій із доказаною ефективністю [2].

1) Щоранку або ж наприкінці дня необхідно присвячувати п’ятнадцять-двадцять хвилин плануванню розпорядку прийдешнього дня. Важливою умовою є те, що додавати до переліку справ необхідно абсолютно усі задачі, навіть якщо вони на перший погляд здаються незначними. Лише таким чином можна застерегти себе від того, що якась справа забудеться.

2) Необхідним аспектом будь-якої задачі є її пріоритетність – від найменш важливих до найбільш вагових – адже саме така ієрархія обумовлює більшу ймовірність того, що найголовніші справи будуть виповнені у першу чергу.

3) Систематичні задачі необхідно трансформувати у звичку, адже якщо виконання таких справ буде сприйматися як повсякденність, необхідність виокремлювати окремий час для них просто пропаде. Отже, процес здобуття цілі буде більш приємнішим.

4) Кожна задача повинна бути оцінена з точки зору того, скільки часу необхідно витратити на її виконання. Це зумовлено тим, що саме чіткі рамки сприяють підвищенню продуктивності людини, а отже, з’являється майже стовідсоткова ймовірність того, що всі дедлайни будуть дотримані.

5) Необхідно використовувати таймер для фіксації часу, адже у противному випадку можна розтягнути робочий процес.

6) У випадку появи задачі, яка виникла досить несподівано, необхідно в обов’язковому порядку додати її до вже існуючого списку із відповідним пріоритетом. Таким чином практикується дисципліна.

7) Під час робочого процесу потрібно лімітувати навколишній простір задля того, щоб мати змогу сконцентруватися на поставленій задачі. Проте, щоб не буди зовсім відірваним від світу, бажано створити «блоки часу» – невеличкий

період під час якого можна передивлятися пошту та повідомлення у соціальних мережах.

8) Важливо пам’ятати про режим праці та відпочинку задля того, щоб уникнути виснаження. Тобто у розклад обов’язково необхідно додати фіксований час для відпочинку.

9) Під рукою завжди повинен бути органайзер (паперовий або електронний, все залежить від особистих уподобань), в якому ведеться облік усіх задач.

1.2 Метод Парето

Правило Парето встановлює, що для більшості процесів, як правило, 80% результатів зумовлено 20% зусиль, що візуально продемонстровано на рис. 1.1. Або ж, іншими словами можна сказати, що невеликий відсоток старань призводить до наслідків, що мають досить вагоме значення.

Рисунок 1.1 – Візуалізація принципу Парето

Правило Парето було розроблено італійським ринковим аналітиком Вільфредо Парето у 1896 році. Саме він зауважив, що на 80% землі в Італії претендує лише 20% населення. Крім того, аналогічні висновки він спостерігав і у своєму розпліднику – 20% рослин приносило 80% врожаю. Тобто таким чином закон Парето отримав і свою другу назву – правило 80/20.

Важливо зазначити, що правило 80/20 – це не формальна числова умова, а скоріш загальна особливість, яка має прояв у фінансових питаннях, бізнесі, ефективному використанні часу та навіть в іграх. Так як нас цікавить правило Парето у тайм-менеджменті, то розглянемо його застосування саме у цій сфері.

Правило Парето можна використовувати задля того, щоб сконцентруватися на тих справах, які необхідно виконати протягом дня. Тобто суть полягає в тому, що із всього переліку справ виконання 20% обов'язків призведе до 80% ефекту, який ви зможете отримати за день [3]. Таким чином, щоб завершити роботу із найбільш продуктивним результатом, необхідно визначити ключові справи та сфокусуватися на них.

Отже, інструкція по використанню правила 80/20 складається із трьох простих пунктів.

1) Першочергово необхідно виконувати найскладніші та найбільш пріоритетні справи. Навіть якщо вам здається, що начебто розібравшись із мілкими задачами, ви зможете більше часу приділити складному завданню, до значних результатів у продуктивності це не призведе.

2) Слідкуйте за головною метою. Необхідно пам'ятати, що всі дрібні задачі потрохи наближують вас до головної цілі, тож потрібно фокусуватися та відстежувати свій прогрес.

3) Визначте фактори, які найбільше відволікають вас від роботи та позбавтесь їх. У першу чергу це стосується смартфона, який слід ставити і беззвучний режим, щоб не звертати увагу на вхідні повідомлення, що не стосуються робочого процесу.

Таким чином, знання правила Парето особливе тим, що воно допомагає у визначенні пріоритетів та сконцентруватися на тих задачах, внаслідок виконання яких буде отримано найбільш вагомий результат.

1.3 Техніка помідору

Техніка помідору була створена у другій половині 1980-х років студентом Франческо Чірілло, який намагався зосередитися на здачі іспитів. Для того, щоб приділити всю увагу виконанню задач, він змусив себе зосередитись на 10 хвилин, і для фіксування часу Франческо використав кухонний годинник у формі помідора. Таким чином і з'явилася дана техніка.

Суть стратегії досить проста та складається із наступних кроків.

1) Складіть список задач на день та підготуйте годинник (можна використовувати смартфон, але за умови, що всі сповіщення будуть вимкнуті).

2) Засікайте 25 хвилин та зосередьтеся на одній справі до тих пір, поки не почуєте сигнал годинника про закінчення відведеного часу.

3) Після закінчення даного блоку часу відмітьте один «помідор» та занотуйте, що ви завершили.

4) Зробіть п'ятихвилинну перерву.

5) Після чотирьох відміток «помідорів» зробіть п'ятнадцяти хвилинну перерву.

Звісно, відрізки часу по 25 хвилин є основою техніки помідора, однак дана практика містить три принципи, за якими можна винести користь з кожного інтервалу.

1) Якщо для виконання завдання потрібно витратити час еквівалентний чотирьом «помідорам», то необхідно розділити цю задачу на більш дрібні етапи. Виконання даного стандарту гарантує неодмінний прогрес у ваших починаннях.

2) Будь-які задачі, які потребують менше часу, ніж тривалість одного «помідора», повинні об'єднатися.

3) «Помідор» є єдиною одиницею часу, яку заборонено порушувати, особливо це стосується перегляду сповіщень або відправки миттєвих повідомлень. Будь-які думки, доручення або прохання необхідно відкласти.

Якщо ж виникли раптові відволікаючі фактори, то допустимо зробити п’ятихвилинну перерву та почати працювати з початку. Досить корисним буде нотування чинників, що відвертають увагу, щоб пізніше проаналізувати їх та попередити виникнення подібних ситуацій.

Під час використання стратегії помідора, людина починає більш адекватно оцінювати обмеження часу та задач загалом, що у свою чергу дозволяє точніше та ефективніше планувати свій день. Чим більше практики – тим краще розвивається здатність розуміти, скільки «помідорів» потребує певна справа.

1.4 Матриця Ейзенхауера

Дана стратегія планування була створена Дуайтом Ейзенхауером, який був 34-м президентом США з 1953 до 1961 року. До того, як він зайняв посаду у владі, Ейзенхауер служив у складі армії Сполучених Штатів у якості Верховного головнокомандуючого Об’єднаних сил під час Другої світової війни. Його службове становище потребували максимальної відповідальності та постійного прийняття важливих рішень, і от під час одного із своїх виступів Ейзенхауер сказав: «У мене є два типи проблем: термінові та важливі. Термінові не є важливими, а важливі ніколи не бувають терміновими» [5]. Більш ніж тридцять років потому Стівен Кові у своїй книжці «7 звичок високоефективних людей» сформував відому всім матрицю на основі знань Ейзенхауера [6].

Матриця Ейзенхауера або ж матриця важливості й терміновості допомагає зосередитися на задачах, базуючись на їх пріоритетності. Завдяки їй можна визначити, які завдання мають бути виконані першочергово, а на які можна навіть не звертати увагу.

Спробувати застосувати матрицю Ейзенхауера варто, якщо людина чітко виражені наступні фактори:

– кожного дня переважає відчуття палаючих дедлайнів, за якими важко встигнути;

– ви зайняті, однак не відчуваєте, що робота виконана ефективно;

– присутні довгострокові цілі, проте відсутній час чи енергія задля їх досягнення;

– маєте складнощі із делегуванням обов'язків або ж не вмієте сказати «ні».

Алгоритм роботи із матрицею Ейзенхауера досить простий: необхідно намалювати свою матрицю та впорядкувати особисті задачі у належний квадрант відповідно до пріоритетності. Усього є чотири сектори, які можна позначити буквами A,B,C,D або цифрами 1,2,3,4, або ж кольорами, як це продемонстровано на рис. 1.2.

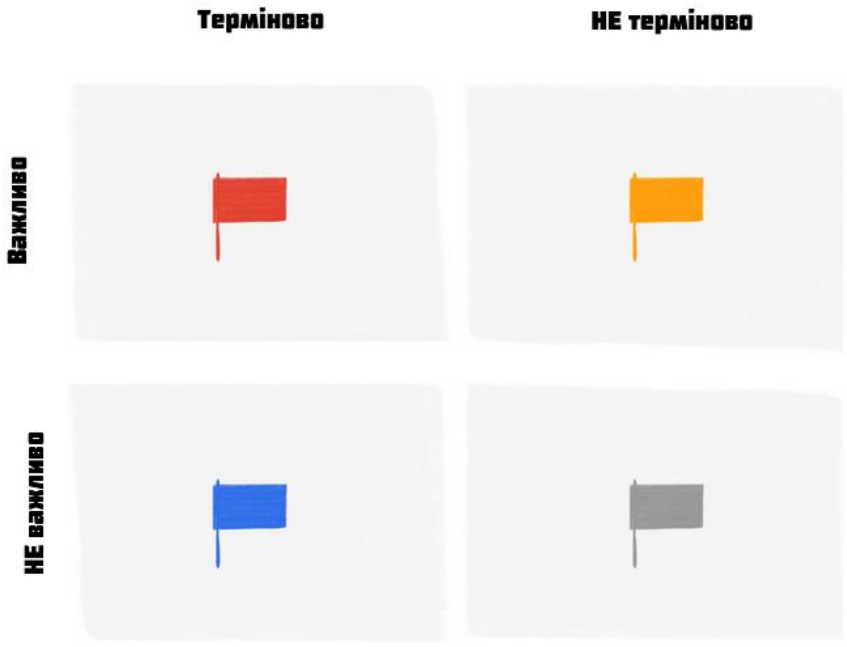

Рисунок 1.2 – Ілюстрація матриці Ейзенхауера

Червоний квадрант: критично важливі задачі, які вимагають швидкого реагування. Як правило, це або ті справи, які раптово виникли із зовнішнього

середовища або ж задачі, що були відкладені до останнього моменту і їхній дедлайн вже зовсім скоро. Прикладом завдань такого типу можуть бути:

- написати розділ курсової роботи до завтрашнього ранку;
- закінчити презентацію для демо проєкту;
- відповісти на листи клієнтів та ін.

Жовтий квадрант: термінові, але у той же час не час не настільки важливі справи, а отже і не вимагають швидкого реагування. Зазвичай це справи, які у перспективі зможуть допомогти в особистому чи професійному плані, наприклад:

- записатися на курс по алгоритмам;
- відвідати джазовий концерт;
- провести рефактор коду для пет-проєкту та ін.

Синій квадрант: критичні справи, які потребують термінового розгляду, однак через те, що вони не є досить важливими, можуть бути делеговані. Прикладами задач даної групи є:

- зібрати інформацію про місцезнаходження членів команди;
- допомогти колегам;
- відповісти на листи із поміткою «терміново» та ін.

Однак важливо зазначити, що делегувати необхідно тоді, коли справи не «підіймаються» до рівня вашої важливості.

Сірий квадрант: мінорні задачі, що не потребують значної уваги та терміновості. Дані завдання можна виконати під час перерви або якщо вам потрібно відпочити від більш вагомих справ. Прикладами можуть бути:

- перевірити акаунт на Facebook;
- обсудити валютний курс із колегами;
- подивитися сонцезахисні засоби в інтернет-магазині та ін.

Звісно, вміння розставляти пріоритети з’являється не одразу, тому при першій спробі скласти матрицю може трапитись таке, що всі справи потраплять

у червоний квадрант. Однак поступово на практиці все поступово буде вирівнюватись та згодом буде набагато легше визначитися із тим, наскільки важливою і терміновою є та чи інша справа.

Висновки до розділу 1

У даному розділі було розглянуто поняття планування задач у цілому та основні принципи, на яких базуються широко розповсюджені стратегії планування часу, а саме правило Парето, стратегія помідора, матриця Ейзенхауера. Всі три методи тайм-менеджменту вимагають певних зусиль та навичок, які можна розвинути лише за допомогою практики.

Крім того, розглянувши особливості найвідоміших методів, можна виділити основні правила, які слід виконувати незалежно від того, яка саме стратегія планування задач застосовується.

1) Завжди занотовувати всі задачі, навіть якщо вони можуть здаватися дрібними на перший погляд.
2) Потрібно визначати пріоритет всіх справ.
3) Реально оцінювати картину часу та не ставити занадто багато об’ємних чи пріоритетних задач на день.
4) Вчитися розбивати завдання на підзадачі.
5) Позбавлятися усіх відволікаючих факторів, особливо вимикати сповіщення на смартфоні.

2 СУЧАСНІ СИСТЕМИ ТАЙМ-МЕНЕДЖМЕНТУ

2.1 Загальні вимоги до розробки систем тайм-менеджменту

Системи тайм-менеджменту підходять як для підприємців та бізнесменів, які бажають керувати своїм часом задля підвищення продуктивності, так і для звичайних людей, що хочуть планувати повсякденне життя.

Однак все ж таки для того, щоб сформулювати основні вимоги, необхідно визначити цільову аудиторію, яка із найбільшою ймовірністю буде використовувати тайм-трекери. До них відносяться:

- великі корпорації;
- малий бізнес;
- стартапи;
- соціальні проєкти;
- команди, що працюють над проєктами;
- фрілансери;
- студенти.

Таким чином системи тайм-менеджменту можуть бути розділені на дві групи: застосунки для підвищення продуктивності бізнесу та персональні планувальники.

Щодо першої категорії, то дані платформи першочергово корисні для будь-яких організацій, в тому числі і для невеликих колективів, і для великих корпорацій. Це зумовлено тим, що більшість працівників не вміє ефективно керувати часом, що у свою чергу призводить до погіршення продуктивності, а це вже у свою чергу погано позначається на графіках продажів.

Отже, от які повинні бути вимоги до програмного забезпечення з планування корпоративного типу.

- Можливість організувати процес керування робочими процесами в одному місці.

– Повна синхронізація даних на пристроях.

– Можливість давати завдання співробітникам, а також контролювати процес виконання цих задач.

– Можливість працювати всією командою над спільним проєктом.

Якщо казати про другу категорію – системи персонального тайм-менеджменту, то їхньою цільовою аудиторією є індивідуальні користувачі: фрілансери, студенти, самозайняті спеціалісти та всі ті, хто бажає підвисити свою продуктивність у повсякденних задачах.

Отже, і вимоги повинні ставитися відповідні.

– Зрозумілий та інтуїтивний інтерфейс для легкого ведення задач.

– Можливість створення задач, які повторюються із певною періодичністю.

– Можливість додавання пріоритетів до завдань.

– Наявність розділу «Вибране».

– Розумна система сповіщень.

– Візуалізація особистої продуктивності.

– Можливість керування задачами.

– Можливість перегляду виповнених задач.

Так як дана робота передбачає створення саме персональної системи планування задач, то всю увагу приділимо саме їм та у наступному підпункті даного розділу розглянемо вже існуючі застосунки.

2.2 Порівняльний аналіз існуючих систем планування задач

Застосунки для планування задач складають досить вагому частину програмного забезпечення, спрямованого на підвищення продуктивності. Відповідно до дослідницької оцінки “Markets and Markets”, індустрія програм для тайм-менеджменту, скоріше за все, досягне 4,33 мільярди доларів до 2023

року [7]. Розглянемо декілька застосунків, які досить широко використовуються у персональному плануванні часу.

Todoist

Todoist – застосунок для складання списку справ та керування задачами як для професійного використання, так і для персонального. За рейтингом користувачів даний сервіс має оцінку 8.9/10 [8].

Сервіс допомагає оптимізувати індивідуальну та командну продуктивність за рахунок поєднання задач, проєктів, коментарів, вкладених матеріалів, сповіщень та іншого функціоналу. У застосунку є два види представлення: у виді дошки (рис. 2.1) або списку (рис. 2.2), що дозволяє керувати проєктами та співпрацювати із іншими членами команди.

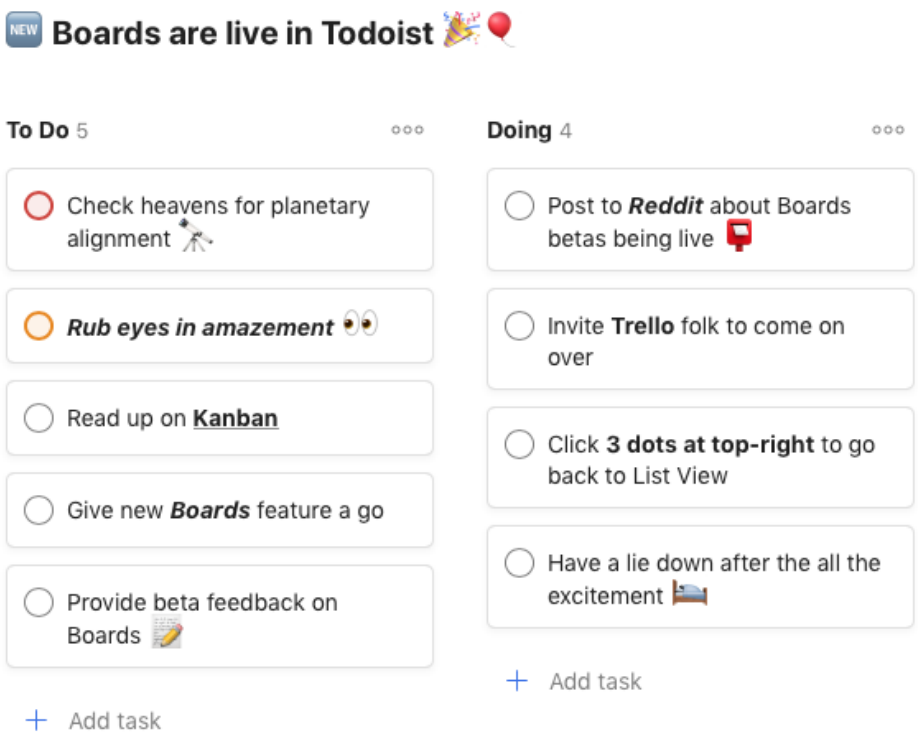

Рисунок 2.1 – Дошки справ у сервісі Todoist

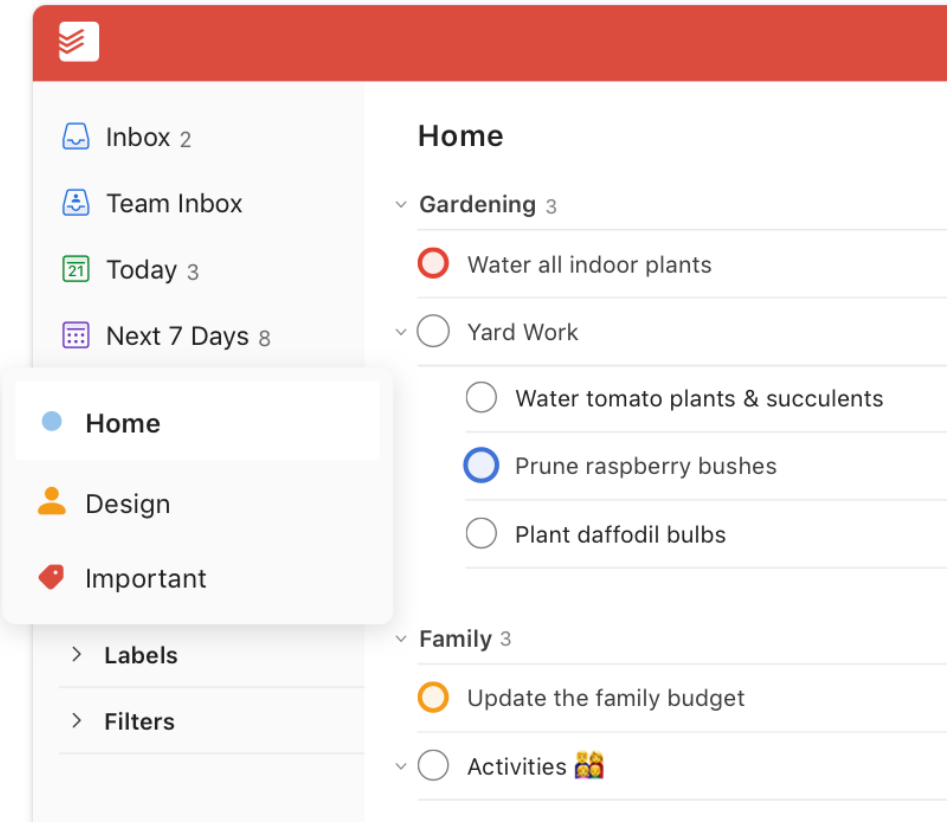

Рисунок 2.2 – Список справ у сервісі Todoist

Кожен користувач може взяти під контроль свою продуктивність за рахунок додавання спеціальних міток, фільтрів та сортувань для створення персоналізованих процедур ефективності. Крім того, у сервісі є можливість налаштування задач, які повторюються, а також виставлення пріоритетів.

Todoist також працює і з сторонніми сервісами, тобто, наприклад, у нього можна інтегрувати Google календар. Застосунок є доступним як і у веб-версії, так і у виді мобільного застосунку, що дозволяє синхронізувати всю інформацію користувача. Єдиним нюансом є те, що безкоштовна версія сервісу надає обмежену кількість функціоналу, для використання усіх привілеїв необхідно оплачувати преміум-підписку.

Trello

Trello – це сервіс із графічним інтерфейсом для управління задачами та проєктами. Незважаючи на те, що як правило, даний застосунок використовується командами розробників програмного забезпечення, він також

може застосовуватись для відслідковування власної продуктивності. Рейтинг сервісу 8.4/10 [8].

Даний сервіс представляє собою цифрову дошку (рис. 2.3), яка дозволяє створювати, керувати та визначати пріоритетність поставлених задач. Крім того, адміністратори мають змогу створювати робочі процеси, визначати задачі для окремих людей або груп, встановлювати дедлайни та відслідковувати прогрес.

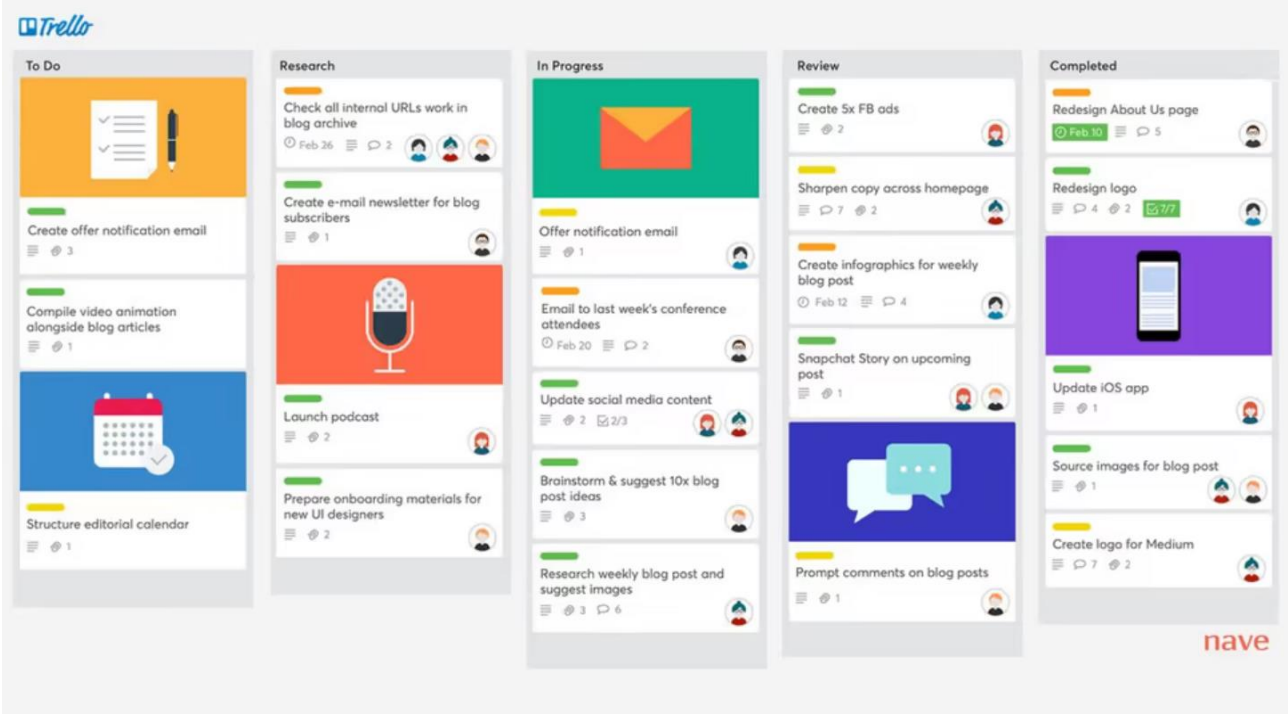

Рисунок 2.3 – Цифрова дошка Trello

Користувачі можуть співпрацювати один із одним за допомогою спеціальних карточок на дошці, а також додавати коментарі й вкладені матеріали до задач. За допомогою цих же карточок можна організувати персональний там-менеджмент із визначенням пріоритетів та строків виконання задач.

Trello пропонує як безкоштовну версію для некомерційного використання, так і бізнес-підписку, яка включає у себе конфіденційність та адміністративний контроль для великих компаній.

Notion

Notion – цифровий робочий простір як для персонального використання, так і для корпоративного. Сервіс дозволяє впорядкувати та керувати нотатками, задачами, проєктами, документами, календарями та багатьма іншими інструментами. Рейтинг застосунку досить високий – 8.9/10 [8].

Дана платформа містить у собі надзвичайно велику кількість функціоналу, який можна персоналізувати під власні потреби. Сервіс надає можливість створення шаблонів різноманітних дошок (рис. 2.4), якими може керувати як і окремий користувач, так і ціла команда. Завдяки доступності на платформах iOS та Android користувачі можуть синхронізувати процеси на всіх своїх особистих пристроях.

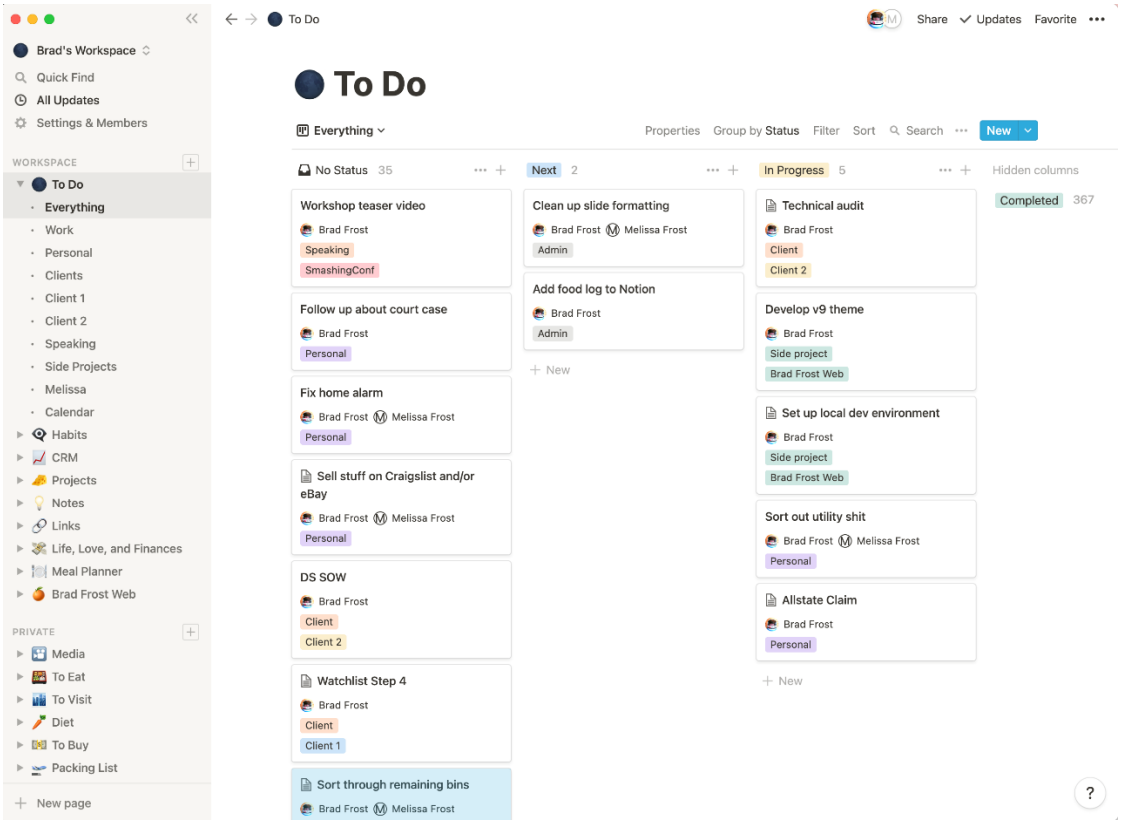

Рисунок 2.4 – Приклад шаблону для списку задач у Notion

Щодо доступності підписки, то у Notion є як безкоштовна версія для персонального користування, так і щомісячна комерційна для бізнесу.

Google Calendar

Google Calendar – це ще один інструмент для тайм-менеджменту, який дозволяє планувати та відслідковувати як різноманітні зустрічі, так і виконання задач різного типу. Рейтинг застосунку складає 9/10 [8].

Сервіс виглядає як звичайний календар (рис. 2.5) дозволяє створювати списки справ та отримувати автоматичні сповіщення про необхідність виконання задач. Користувачі мають змогу продивлятися розклад у різноманітних формах: щомісячно, щотижнево, щоденно. Крім того, сервіс підтримує планування групових заходів і також надає можливість продивлятися розклад інших у відкритому чи закритому режимах.

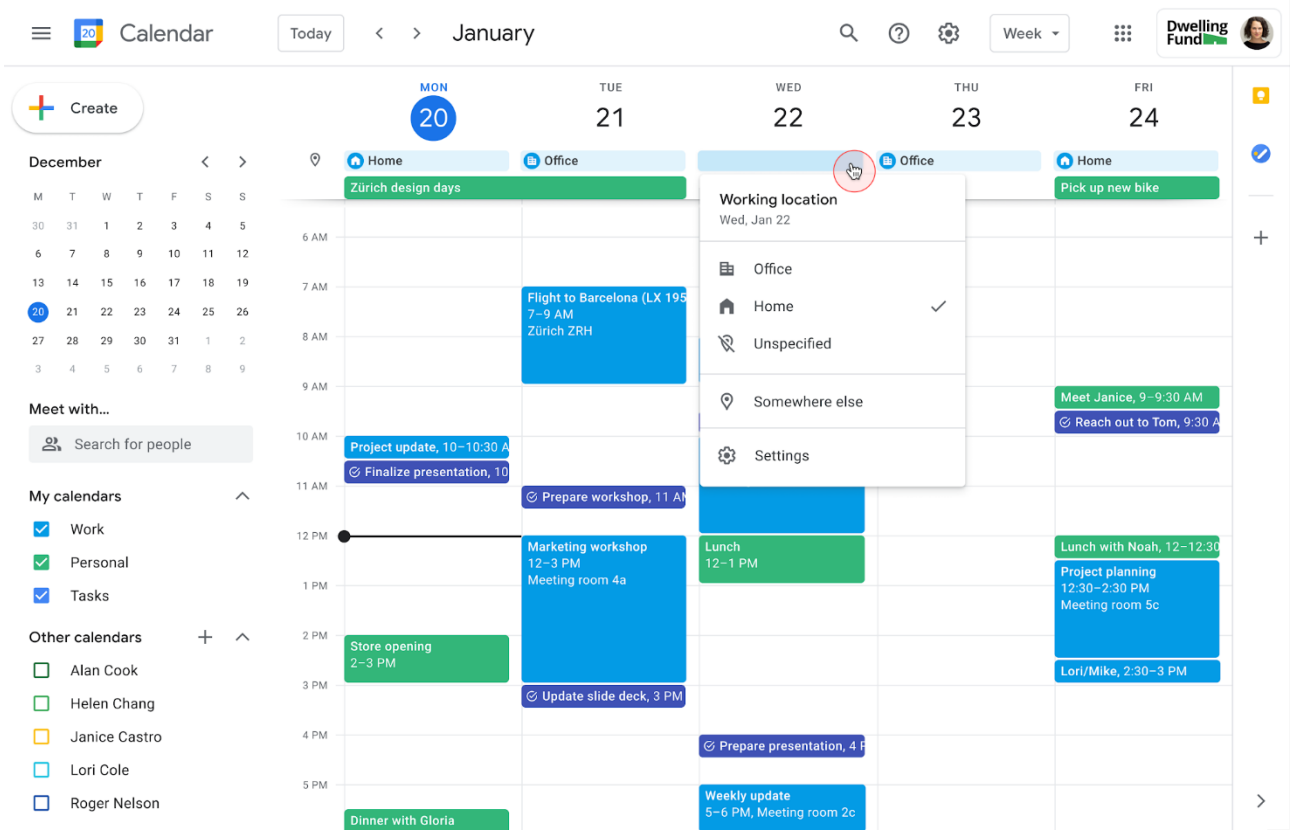

Рисунок 2.5 – Інтерфейс сервісу Google Calendar

Великою перевагою застосунку є те, що він синхронізується із аккаунтом Gmail, що дозволяє автоматично отримувати сповіщення на пошту. Крім того

для синхронізації є застосунки на платформи iOS та Android. А також варто зазначити, що наразі сервіс є безкоштовним.

2.3 Огляд засобів для розробки

У 2022 році є декілька лідерів мов програмування для розробки веб-застосунків, серед яких JavaScript, Python, Java, C#, PHP та TypeScript. Для того, щоб обрати той варіант, який найбільш всього підходить, необхідно розглянути всі.

JavaScript досить довгий час був однією з найпопулярніших мов для веб-розробки. І насправді дана мова зберігає свої позиції як мова програмування, що найбільш широко використовується, протягом останніх 9 років. Близько 64,96 відсотка професійних розробників програмують на JavaScript, якщо керуватися результатами опитування розробників Stackoverflow Developer Survey Results 2021 [9]. Насправді є ціла низка причин, яка робить JavaScript настільки популярним.

По-перше, дана мова програмування – це сильна та гнучка мова, яка постійно впроваджується в різні веб-браузери. Крім того, JS може використовуватися як у клієнтській, так і у серверній веб-розробці, що дозволяє їй працювати на будь-яких пристроях – від веб-браузерів до потужних серверів.

По-друге, JavaScript пропонує велику кількість бібліотек фреймворків, які дозволяють програмістам створювати складні програми з мінімальними накладними витратами. А також завдяки своїй простоті менеджери пакетів JavaScript дозволяють розробникам просто і швидко поширювати свій код між командами, роблячи процес комунікації гладкішим. І останнє, але не менш важливе: з появою Node.js у 2009 році розробники отримали можливість писати код однією мовою, що дозволяє їм мати єдину мову для фронт- та бек-ендів.

Python є не менш популярною мовою для веб-розробки. Це можна пояснити зростаючою популярністю машинного навчання серед розробників, які вивчають Python через його простоту використання та здатність враховувати

численні парадигми програмування. Python надає широкий вибір програмних бібліотек, які можна вставити в код і виростити до величезних програм, що дає широкі можливості для розробки, від онлайн та настільних додатків до системної діяльності. Через свій англомовний синтаксис він зазвичай рекомендується для новачків у розробці програмного забезпечення. У порівнянні з іншими мовами веб-застосунків, вивчення та розробка коду займає значно менше часу.

Java, безсумнівно, є найпопулярнішою платформою для розробки корпоративних програм завдяки своїм розширеним можливостям безпеки, які є обов'язковою умовою для будь-якого бізнесу. Java також забезпечує простий, гнучкий і багаторазово використовуваний код, який перетворюється на зрозумілий інтерфейс для додатків. Java - одна з найбільш ідеальних мов для розробки корпоративних програм, оскільки на ній доступні мільйони бібліотек. Ще однією перевагою Java є високий рівень свободи платформи завдяки можливостям віртуальної машини JVM, що є середовищем виконання для багатьох відомих мов програмування, включаючи Kotlin, Scala та Groovy. Java оновлюється кожні шість місяців, незалежно від лих або пандемій, що зміцнює його репутацію як надійної та гідної довіри мови.

C# – це свого роду більш безпечна та швидка версія C та C++. Дана мова залишається однією з найпопулярніших мов програмування веб-застосунків через два десятиліття після своєї появи. Від серверних програм до мобільних ігор – ця мова використовується для широкого спектру завдань. Це об'єктно-орієнтована, пряма мова з виконуваним кодом та налаштуваннями часу виконання, оптимізована для CLI. Функція безпеки типів запобігає помилкам розробника під час компіляції та виконання, а функція автоматичного складання сміття видаляє та стирає все сміття. C# заснований на CRL, що дозволяє легко інтегрувати його з компонентами, розробленими іншими мовами. Велика кількість доступних бібліотек спрощує процес розробки та робить простий реалізацію широкого спектра функцій. C# продовжує залишатися популярною

мовою для вивчення та потужним гравцем на ринку завдяки значному обсягу документації та постійним інвестиціям з боку Microsoft.

PHP все ще продовжує використовуватись на багатьох веб-сайтах, навіть незважаючи на те, що велика кількість розробників вважає цю мову застарілою та не виявляє бажання працювати із проєктами, написаними нею.

PHP – це мова з відкритим вихідним кодом для створення динамічних та статичних веб-сайтів, оскільки він поставляється з величезною кількістю вбудованих утиліт та додаткових модулів. Оскільки PHP має довгу історію, він користується перевагами великої спільноти користувачів, які створили фреймворки, бібліотеки та інструменти автоматизації, щоб зробити процес розробки швидше та простіше. PHP має низьку криву навчання та простий у освоєнні для новачків. Він використовується для швидкого створення високоякісних веб-застосунків, які легко підтримувати і усувати неполадки в майбутньому. Незважаючи на жорстку конкуренцію з боку вищезгаданих мов, попит на інженерів PHP, як і раніше, високий.

В останні кілька років Typescript стає все більш популярним. Крім підтримки Microsoft і Google існує ще кілька факторів, що сприяють популярності TS. TypeScript розширює JavaScript, надаючи потужніші інструменти в будь-якому масштабі. Простіше кажучи, TypeScript – це JavaScript зі статичною типізацією. TypeScript може виявити досить значну кількість проблем JavaScript. Код на TS легше переробляти, ніж на JS, що дозволяє розробникам швидше усувати помилки і переписувати код. Крім того, код, написаний на C#, Java або PHP, набагато легше перетворити на TS, ніж код, написаний на JS.

Враховуючи все вищеописане, вважаю найбільш доречним використовувати найбільш популярну та гнучку мову програмування – JavaScript, а саме один із її найбільш популярних фрейворків – React JS.

React JS – ефективна та гнучка бібліотека із відкритим початковим кодом, яка основа JavaScript та використовується для створення простих, швидких та масштабованих інтерфейсів веб-застосунків (рис. 2.6)

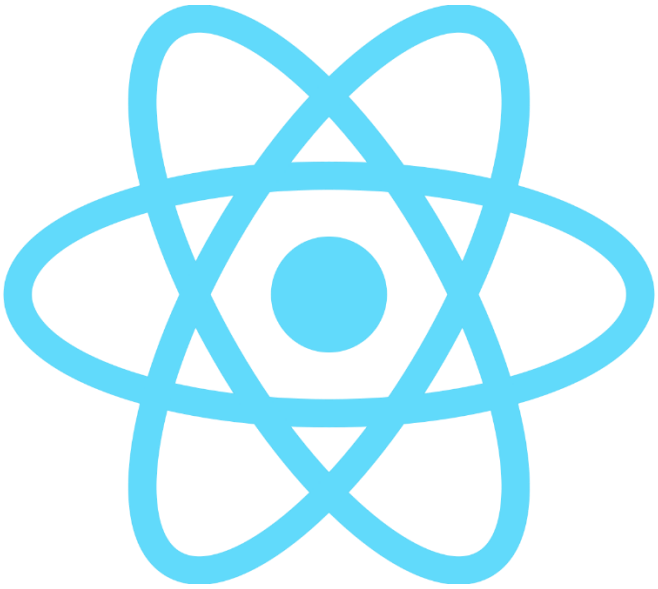

Рисунок 2.6 – Логотип бібліотеки React JS

Із самого моменту створення React JS був і залишається найбільш поширеним інструментом у фронтенд-розробці. Згідно до останнього дослідження Stack Overflow, React JS є найпопулярнішим веб-фреймворком, в той час як Angular та Vue займають лише 4 та 5 місце відповідно [9].

У сучасній розробці React JS використовує близько 10630448 веб-сайтів [10], серед яких такі гіганти індустрії, як Apple, Netflix, Paypal та інші.

Важливою особливістю React-застосунків є те, що вони являють собою SPA або односторінкові застосунки. Тобто проєкт містить лише один HTML-файл – *index.html*.

Односторінковий застосунок – це застосунок, який повністю працює у браузері та не потребує перезавантаження сторінки. У сучасному інтернет-просторі майже кожна система використовує саме SPA. Найбільш відомими прикладами є такі веб-сайти, як Gmail. Google Maps, Netflix, Facebook та інші.

SPA посилає запит на розмітку та дані окремо, а потім відмальовує сторінки безпосередньо у браузері. Передові JavaScript-фреймворки, такі як Angular, React, Vue та інші, дозволяють створювати застосунки саме такого типу.

Перевагами SPA є:

– швидкість, адже більшість ресурсів (HTML, CSS та різноманітні скрипти) завантажуються лише один раз;

– оптимізований та спрощений процес створення, адже для рендерингу сторінок на сервері немає потреби створювати код;

– легко відстежувати мережеві операції, перевіряти елементи сторінок та аналізувати дані, пов’язані із додатком, за допомогою інструментів сучасних браузерів;

– мають змогу ефективно кешувати будь-яке локальне сховище, адже програма надсилає лише один запит, зберігає всі дані, а потім використовує їх для роботи навіть за відсутності підключення до інтернету.

Протилежним до SPA є MPA або багатосторінкові застосунки. Тобто це застосунки, в яких кожна зміна відображення даних або надсилання даних на сервер, вимагає, щоб браузер створював нову сторінку із сервера. Ці програми мають кілька рівнів інтерфейсу користувача через велику кількість матеріалу. В результаті це додає більше складності та робить розробку більш складною, ніж односторінкова програма.

Веб-застосунок планування задач буде використовувати модель SPA, адже це є одним із ключових принципів бібліотеки React, який крім цього має ще достатню кількість переваг, які вирізняють його серед інших фреймворків.

1) Швидкість. React дозволяє розробникам використовувати певні компоненти застосунку як на стороні клієнта, так і на сервері, що у свою чергу пришвидшує процес розробки. Кажучи простими словами, програмісти можуть писати різні області програми і жодна з модифікацій не вплине на логіку застосунку.

2) Гнучкість. Завдяки своїй модульній структурі код, написаний за допомогою React набагато легше підтримувати, а також він є досить універсальним у порівнянні із іншими фреймворками. Саме завдяки такій гнучкості економиться значна кількість часу та ресурсів, а відповідно і грошей.

3) Продуктивність. Ключовими особливостями фреймворку є віртуальний DOM та рендеринг на стороні сервера, що дозволяє виконувати складні проєкти значно скоріше.

4) Зручність використання. За умови базового знання JS розгорнути проєкт на React JS досить просто.

5) Повторне використання компонентів. Саме це є однією із головних переваг фреймворку, адже завдяки можливості повторного використання розробники економлять час. Крім того, будь-які зміни, що будуть внесені в одну частину застосунку, ніяким чином не вплинуть на інші.

Якщо ж казати про недоліки даної бібліотеки, то можна виділити наступне.

1) Недостатнє покриття документацією. Через те, що React JS досить активно та часто оновлюється, розробники не завжди встигають покрити усі моменти.

2) Швидкість розробки. Незважаючи на те, що даний момент був описаний у якості привілеї даної бібліотеки, він також є і негативним у тому сенсі, що активне оновлення React JS вимагає постійного навчання, що може обурювати розробників.

У будь-якому випадку плюси даного фреймворку все ж таки переважають, і тому він не втрачає своєї популярності, а навіть навпаки, збільшує її із кожним роком.

Ще одним важливим аспектом розробки із допомогою React є state (або стан). Об’єкт стану – це вбудована сутність React, яка зберігає дані або інформацію про компоненту. Цей стан може змінюватись час віл часу і внаслідок чого спричиняти повторний рендер компонента із новими даними. Крім того, state буває локальний та глобальний. Перший використовується суто в окремій

компоненті, а от другий дозволяє легко прокинути дані у будь-яке місце проєкту без зайвих маніпуляцій.

Компоненти веб-застосунку повинні «спілкуватися» між собою глобально, як мінімум, для того, щоб всі частини проєкту мали доступ до даних користувача. Для цього необхідно використати бібліотеку для стейт-менеджменту. Найбільш популярною у зв'язці із React JS є Redux [11].

Redux – сполучний шар між React UI та об'єктом глобальної істини (рис.2.7). Він дає змогу компонентам React брати дані із сховища Redux та відправляти туди actions (дії) для оновлення стану.

Рисунок 2.7 – Логотип Redux

Найбільшою перевагою використання зв'язки React-Redux є оптимізація застосунку. Завдяки системі стейт-менеджменту компонент не повинен передавати велику кількість даних своїм дочірнім компонентам, цим всім займається Redux. Даний процес можна побачити на рисунку 2.8.

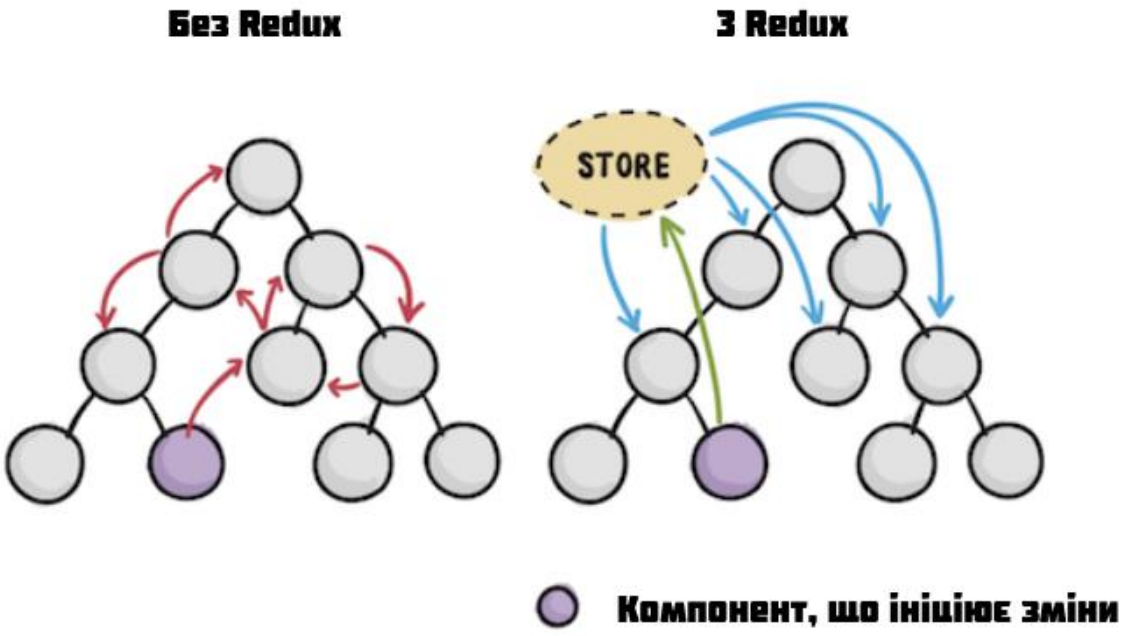

Рисунок 2.8 – Ілюстрація роботи із використанням Redux та без нього

Крім того, так як Redux розвивається майже настільки ж швидко, наскільки і React. І для більш простого використання було створено бібліотеку Redux Toolkit [12]. Вона основана на ядрі Redux та надає API для спрощення типових випадків використання глобального сховища. Саме тому у проєкті буде використовуватися RTK.

Останнім засобом розробки, про який необхідно подумати заздалегідь – база даних. Загалом існує досить багато типів баз даних, які можуть використовуватися в залежності від потреб продукту [13]:

- прості структури даних (csv-файли);
- ієрархічні БД (файлові системи, DNS);
- мережеві БД (IDMS);
- реляційні БД (MySQL, PostgreSQL);
- БД NoSQL (Redis, Firebase);
- документоорієнтовані БД (MongoDB, RethinkDB);
- графові БД (Neo4j, Dgraph);

– стовпчикові БД (Cassandra, HBase);
– БД часових рядів (TimescaleDB, OpenTSDB);
– newSQL БД (Calvin, VoltDB);
– багатомодельні БД (ArangoDB, Couchbase).

Для створення веб-застосунку планування задач буде використано NoSQL базу даних, яка у свою чергу представляє собою нетабличну БД. В залежності від моделі даних вони можуть зберігати такі типи, як документ, ключ-значення, граф та інші. Крім того, дані БД мають адаптовані схеми, за рахунок яких можуть досить легко обробляти великі обсяги даних, а також справлятися із високим навантаженням користувачів.

Головними перевагами NoSQL баз даних є:

– гнучкість (підходять як для напівструктурованих даних, так і для неструктурованих);
– масштабованість (розширюються не за допомогою додавання дорогих та надійних серверів, а завдяки використанню розрізнених кластерів устаткування);
– висока продуктивність (БД NoSQL адаптовані під конкретні моделі даних і шаблони доступу, що дозволяє працювати краще, ніж реляційним баз даних під час виконання аналогічних завдань);
– високофункціональні API та типи даних (пропонують високофункціональні API та типи даних, спеціально розроблені для кожної моделі даних).

Однією із найбільш доступних баз даних типу NoSQL є Firebase. Firebase Realtime Database – це хмарна база даних NoSQL, яка дозволяє зберігати та синхронізувати дані в режимі реального часу між вашими користувачами.

Realtime Database поставляється з SDK для мобільних та веб-застосунків, що дозволяє створювати програми без використання серверів. Крім того, Firebase пропонує Cloud Functions for Firebase для запуску бекенд-коду, який реагує на

події, що створюються вашою базою даних. Однак дана послуга вже є доступною у платному тарифі.

Висновки до розділу 2

В даному розділі було визначено та описано загальні вимоги до систем тайм-менеджменту, як корпоративних, так і персональних. Крім того, було проаналізовано існуючі застосунки планування задач. Також було розглянуто популярні мови програмування для веб-розробки і бібліотеку React JS для створення користувацьких веб-інтерфейсів, визначено її переваги та недоліки. Було розглянуто Redux як систему стейт-менеджменту застосунку і обґрунтовано вибір бази даних для реалізації веб-застосунку.

3 РОЗРОБКА ВЕБ-СИСТЕМИ ПЛАНУВАННЯ ЗАДАЧ

3.1 Проєктування інформаційної системи

Веб-застосунок повинен надавати користувачу можливість створення та керування власними задачами із функціональністю створення персональних проєктів, а також планування групових заходів.

Проєкт – контейнер, який допомагає сортувати задачі користувача за призначенням. Проєкти створюються безпосередньо користувачем. При цьому одна задача може належати лише одному проєкту. Кожен проєкт складається із таких атрибутів:

- назва (обов’язково);
- опис (необов’язково);
- колір (необов’язково, але за замовченням йде колір теми).

Задача – чітко сформульована користувачем справа, яку необхідно виконати. Причому кожна задача містить певні обов’язкові та необов’язкові атрибути, серед яких є наступні:

- назва (обов’язково);
- назва проєкту (необов’язково);
- дедлайн (необов’язково);
- час виконання у розмірності «помідорів» (необов’язково);
- електронна пошта користувача, із яким планується подія (необов’язково).

Особливістю веб-застосунку є імплементація розмірності часу у «помідорах». Для цього буде виділено окрему сторінку із налаштуваннями таймера. За замовченням на робочу концентрацію буде виділено 25 хвилин, на коротку перерву – 5 хвилин, на довгу – 15. Однак за бажанням користувач зможе регулювати ці часові рамки.

Планування групових заходів буде виглядати таким чином: на окремій панелі календаря користувач зможе ввести пошту людини, із якою він би хотів спланувати спільну подію та система відобразить «вікна». На базі цього вільного часу вже можна створювати подію.

3.2 Опис програмної реалізації застосунка

Для початку створюємо проєкт за допомогою create-react-app [14] та підготовлюємо необхідну структуру (рис. 3.1)

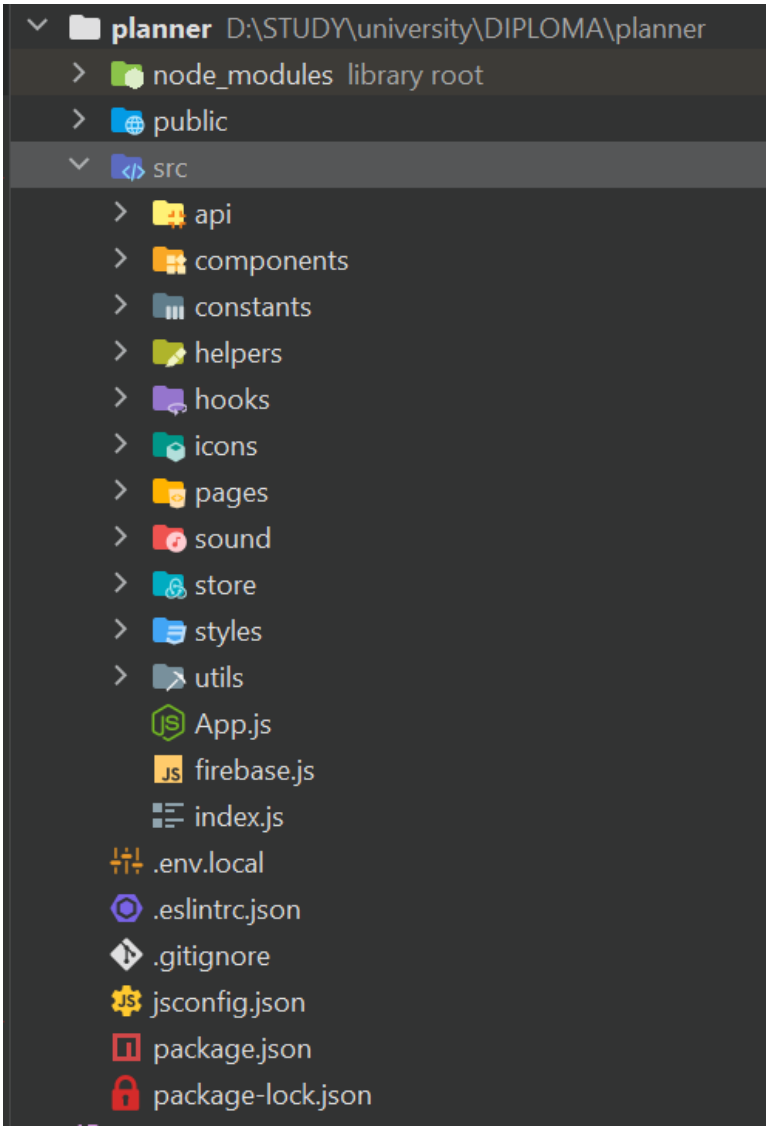

Рисунок 3.1 – Структура проєкту

Тобто таким чином для чіткої структуризації маємо наступні директорії:

– api: містить файли HTTP-клієнта (в даному проєкті – axios);

– components: містить основні компоненти, які в свою чергу формують контейнери для сторінок застосунку;

– constants: усі константи (повідомлення про помилки, інформаційні сповіщення та інше);

– helpers: містить прості допоміжні функції, що можуть використовуватися у хуках або компонентах;

– hooks: зберігаються всі кастомні хуки (ті, які були створені самостійно);

– icons: фали із всіма іконками;

– pages: компоненти, що сформовані завдяки компонентам із *components*;

– store: містяться схеми глобального сховища (у даному випадку Redux);

– styles: стилі застосунку;

– utils: допоміжні компоненти.

Наступним кроком буде установка модулів, які будуть використовуватися при створенні веб-застосунку.

По-перше, у якості глобального сховища застосунку буде використовуватися Redux, а для управління ним – інструмент Redux Toolkit, який дозволяє значно спростити механізм керування даними. Також для відлагодження застосунку буде використано проміжне програмне забезпечення – redux logger, що дає можливість відслідковувати зміни стану у консолі розробника.

По-друге, для стилізації веб-застосунку буде використано Styled Components, які дозволяють використовувати стилі у вигляді компонентів. Тобто

таких підхід CSS-in-JS об’єднує JavaScript та CSS для створення стилізованих компонентів.

По-третє, у якості бази даних буде використано Firebase – NoSQL БД, що розміщується в хмарі і дозволяє зберігати та синхронізувати дані користувачів у реальному часі.

Тепер встановлюємо допоміжні бібліотеки:

- moment: робота із датами у JS;
- react calendar: календар для React-застосунків;
- react countdown circle timer: таймер зворотного відліку із можливістю кастомізації;
- uuid: генерація UUID.

Ще одним підготовчим моментом є конфігурація ESLint, який допомагає дотримуватись певних стильових норм написання коду. Крім того, лінтери корисні у тому, що завдяки ним можна виявити специфічні типи дефектів, наприклад, пов’язаних із областю видимості змінних. Прикладами таких помилок є присвоєння неоголошеним змінним або ж використання невизначених змінних.

Налаштування лінтера – досить простий процес. Потрібно встановити потрібний модуль (в нашому випадку eslint), а після цього додати до проєкту конфігураційний файл *eslint.json*, у якому будуть описані правила написання коду для даного проєкту. Наприклад, якщо ми хочемо виводити застереження при використанні наявності змінних, що не використовуємо, необхідно прописати наступне: "no-unused-vars": "warn".

У даному проєкті буде використовуватися вже готова шаблонна конфігурація лінтера – eslint-config-airbnb.

Останнім підготовчим кроком буде задання налаштувань у файлі *jsconfig.json*, а саме прописуємо правило, яке дозволить імпортувати компоненти за абсолютним шляхом:{ "compilerOptions": { "baseUrl": "src" } }.

Після встановлення необхідних модулів у файлі *firebase.js* описуємо конфігурацію БД (рис. 3.2). Причому всі необхідні дані для підключення зберігаємо у файлі *env.local*, і звертаємося до них за допомогою *process.env*.

```
import firebase from 'firebase/compat/app';
import 'firebase/compat/auth';
import 'firebase/compat/firestore';

export const app = firebase.initializeApp( options: {
  apiKey: process.env.REACT_APP_FIREBASE_API_KEY,
  authDomain: process.env.REACT_APP_FIREBASE_AUTH_DOMAIN,
  projectId: process.env.REACT_APP_FIREBASE_PROJECT_ID,
  storageBucket: process.env.REACT_APP_FIREBASE_STORAGE_BUCKET,
  messagingSenderId: process.env.REACT_APP_FIREBASE_MESSAGING_SENDER_ID,
  appId: process.env.REACT_APP_FIREBASE_API_ID,
});
```

Рисунок 3.2 – Конфігурація Firebase

Наступним кроком є написання захищеного роуту, який необхідний для того, щоб неавторизовані юзери не мали змоги передивлятися сторінки, що призначені для безпосередніх користувачів. В даному проєкті використано реалізацію, у якій авторизація перевіряється наявністю відповідного запису у localStorage і вже базуючись на ці дані, відбувається перенаправлення користувача або на потрібну сторінку, або ж за кореневим роутом (в нашому випадку це сторінка логіну). Програмний алгоритм роботи захищеного роута зображено на рис. 3.3.

```
import React from 'react';
import { Route, Redirect } from 'react-router-dom';

const isAuth = () => localStorage.getItem( key: 'USER_TOKEN');

const SecuredRoute = ({ children, ...params }) => (
  <Route
    {...params}
    render={({ location : Location<LocationState> }) => (isAuth() ? (
      children
    ) : (
      <Redirect to={{ pathname: '/', state: { from: location } }} />
    ))}
  />
);

export default SecuredRoute;
```

Рисунок 3.3 – Компонент SecuredRoute

Тепер у файлі із роутінгом застусонку усі компоненти, які мають бути доступними лиже авторизованим користувачам, будуть обгорнуті у компонент SecuredRoute, а загальнодоступний компонент логіну буде обгорнутий за допомогою Route із react-router-dom (рис. 3.4).

```
<SecuredRoute path="/pomodoro">
  <Pomodoro />
</SecuredRoute>
<Route exact path="/">
  <Login />
</Route>
```

Рисунок 3.4 – Приклад використання захищеного роута

Загалом веб-застосунок буде містити 7 головних компонентів (сторінок), з яких 2 є публічними (логін та реєстрація) та 5 – захищеними. Спрощена схема проєкту представлена на рис. 3.5.

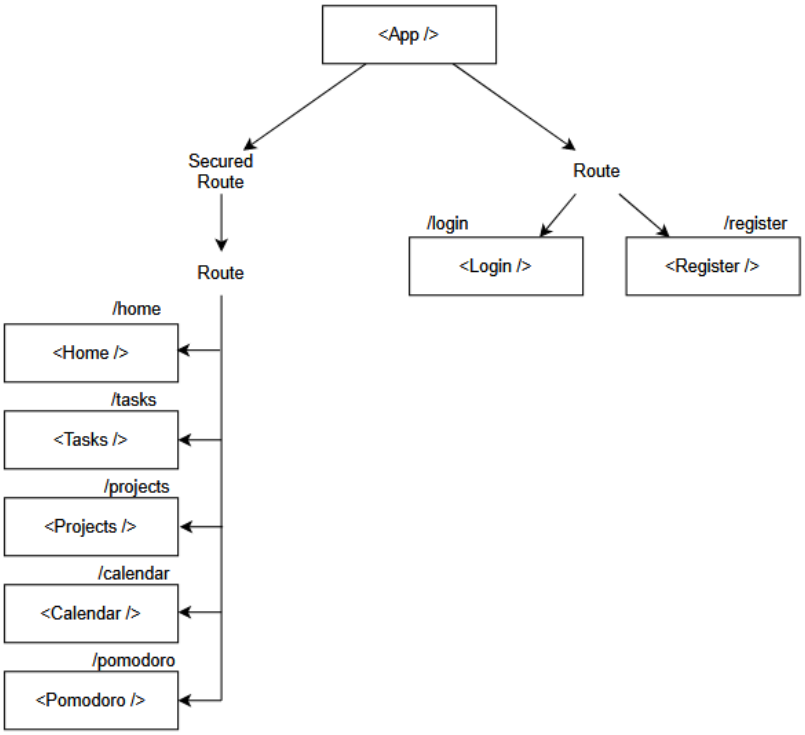

Рисунок 3.5 – Схема роутів застосунку

Для реалізації авторизації та реєстрації було використано вбудований механізм Firebase, який дозволяє обрати саме той метод (або методи), які найкраще підходять для застосунку (рис. 3.6).

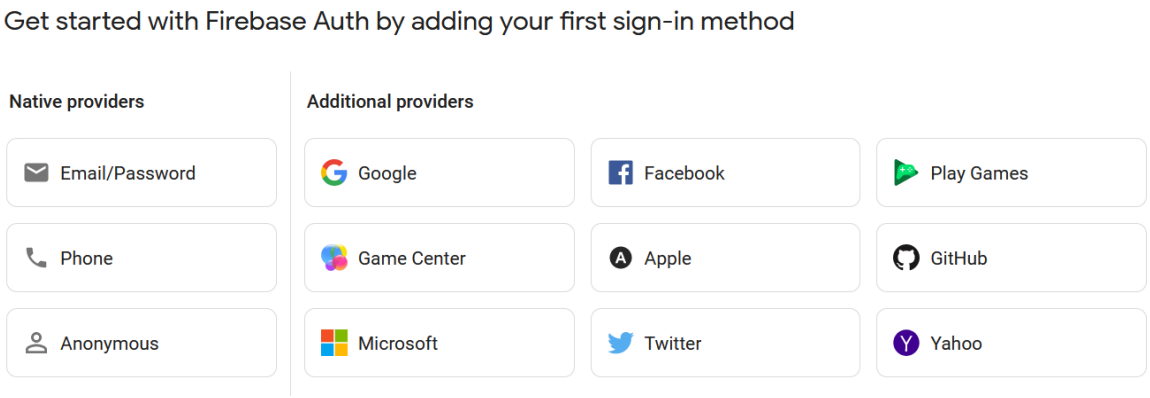

Рисунок 3.6 – Доступні методи авторизації за допомогою Firebase

Для даного веб-застосунку планування задач будемо використовувати метод із електронною поштою та паролем. Лістинг коду реалізації методів авторизації та реєстрації за допомогою Firebase знаходяться у Додатку А.

Таким чином в нас з'являється перша сутність застосунку – юзери, які за структурою містять ім'я, електрону пошту, пароль, унікальний ідентифікатор (необхідний для оновлення даних) та токен (для аутентифікації). Дві інші сутності, необхідні для коректного функціонування застосунку – задачі та проєкти. Для кожної із сутностей необхідно написати service API, а також redux slice. Так як вони практично однотипні (різниця буде лише у полях), у додатку Б знаходиться приклад API для сутності користувача, а у додатку В – приклад частини глобального сховища.

Веб-застосунок буде складатися із п'яти сторінок, які міститимуть певний функціонал. У свою чергу для полегшення розробки кожна сторінка буде розбита на частини (модулі), які можна легко повторно використати та/або за необхідності розширити функціонал.

Одним із прикладів розбиття на мілкі частини є компонент TaskItem, який відповідає за безпосереднє відображення задачі. Винесення саме цього компоненту є важливим, адже він буде зустрічатися у декількох батьківських компонентах, які у свою чергу відображатимуть список задач (рис. 3.7).

Рисунок 3.7 – Наглядна кількість викликів компоненти TaskItem

Кожен компонент у свою чергу розміщується у окремій папці із структурою, відображеною на рисунку 3.8. Тобто є файл *index.js*, який відповідає лише за експорт компонента. Файл із розширенням *.js* відповідає за саму компоненту, її логіку та відображення. І останній файл *.styles.js* – стилізація компонента. Звісно, якщо необхідні допоміжні компоненти, вони також будуть розміщені у папці.

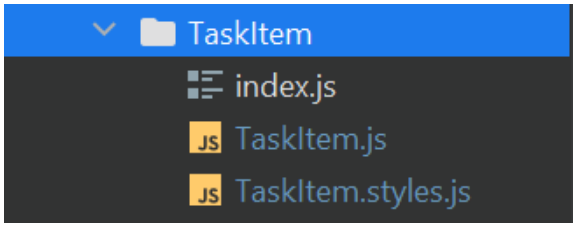

Рисунок 3.8 – Структура папки окремого компонента

Крім того, такі маленькі модулі, як TaskItem складають більш великі структури, як TasksList. На рисунку 3.9 відображено папки із усіма модулями, які складають компонент списку задач. Також обов'язково присутній файл *index.js*, завдяки якому імпорти будуть виглядати красивіше.

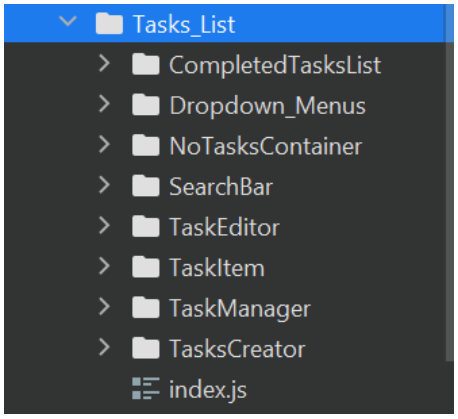

Рисунок 3.9 – Структура папки для списку задач

Таким чином, із маленьких частинок поступово пишемо проєкт та створюємо такі батьківські компоненти, як Calendar, Home, Pomodoro, Projects, Tasks.

Готовий проєкт запускається локально на http://localhost:3000/home за допомогою команди yarn start.

3.3 Користувацький інтерфейс

Головним екраном застосунку для неавторизованого користувача є форма входу (рис. 3.10), на якій присутні два поля: електронна пошта та пароль. У разі введення облікових даних, що не існують у базі даних, користувач отримає відповідне повідомлення угорі форми (рис. 3.11)

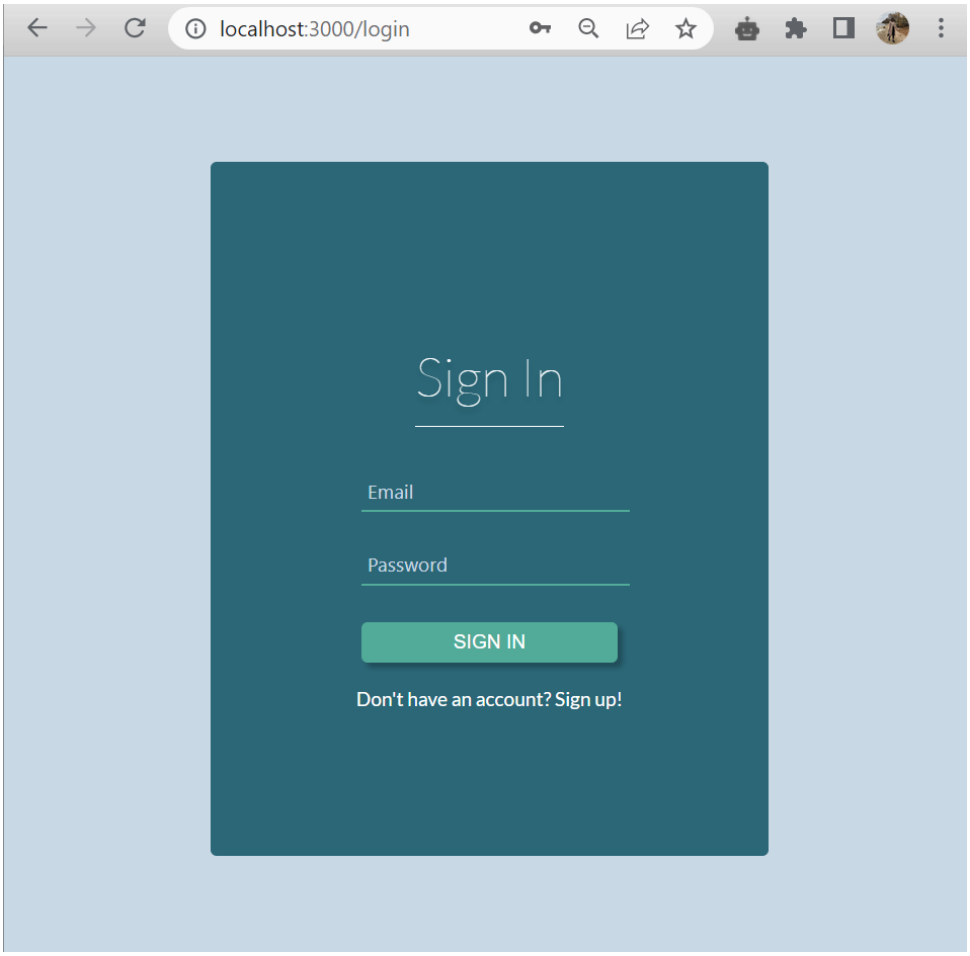

Рисунок 3.10 – Форма входу до веб-застосунку

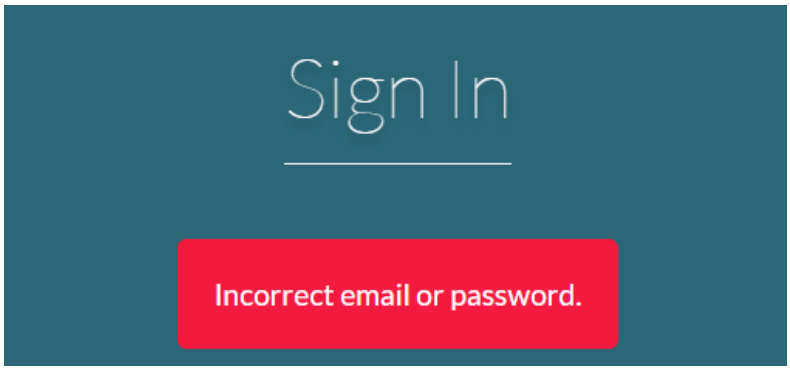

Рисунок 3.11 – Повідомлення про введення некоректних даних

Якщо користувач немає облікового запису, він може перейти за посиланням внизу форми на сторінку реєстрації (рис. 3.12). Для того, щоб зареєструватися, необхідно ввести своє ім’я , електронну адресу та пароль. Кожне із полів має валідацію на те, що воно заповнене, а також на відповідність формату. Електрона пошта повинна відповідати regex шаблону \S+@\S+\.\S+ (тобто звичайний вигляд пошти типу test@test.com), а пароль – (?=.{8,}) (тобто містить не менше восьми символів). У разі невідповідності якогось із критеріїв користувач отримує повідомлення про помилку аналогічне до того, що проілюстровано на рисунку 3.11, але із відповідним текстом помилки.

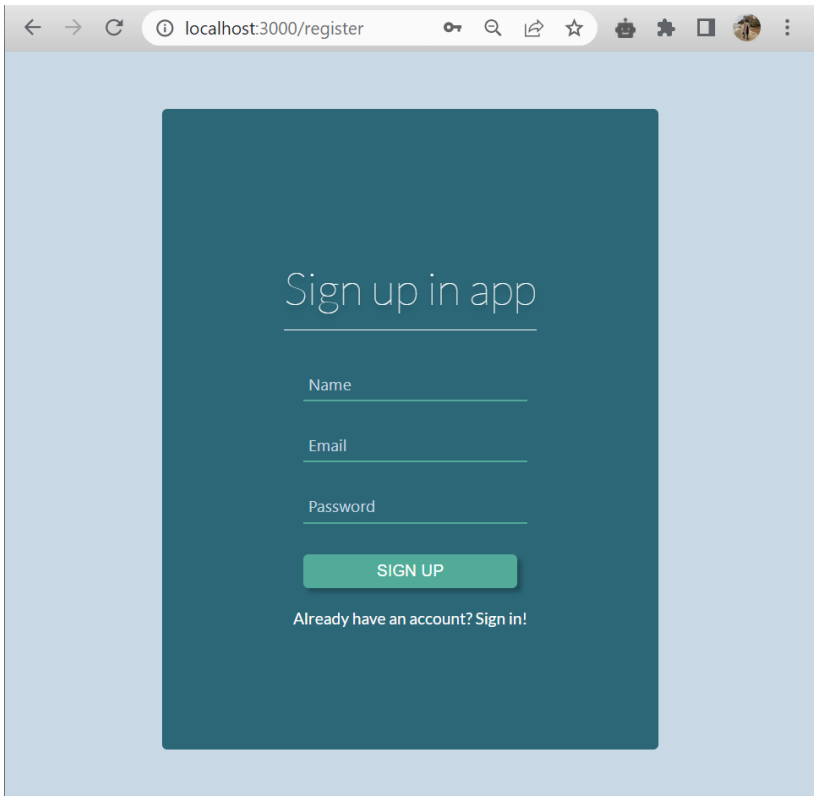

Рисунок 3.12 – Форма реєстрації нового користувача

Після реєстрації та успішного входу до створеного акаунту користувач потрапляє на головний екран (рис. 3.13). У нижній частині відображається панель навігації між сторінками, а справа вгорі присутня кнопка виходу із облікового запису. У разі відсутності вибраних задач, на головному екрані буде відображатись відповідне повідомлення. Як тільки будуть додані задачі – з’являться картки із їх відповідним відображенням.

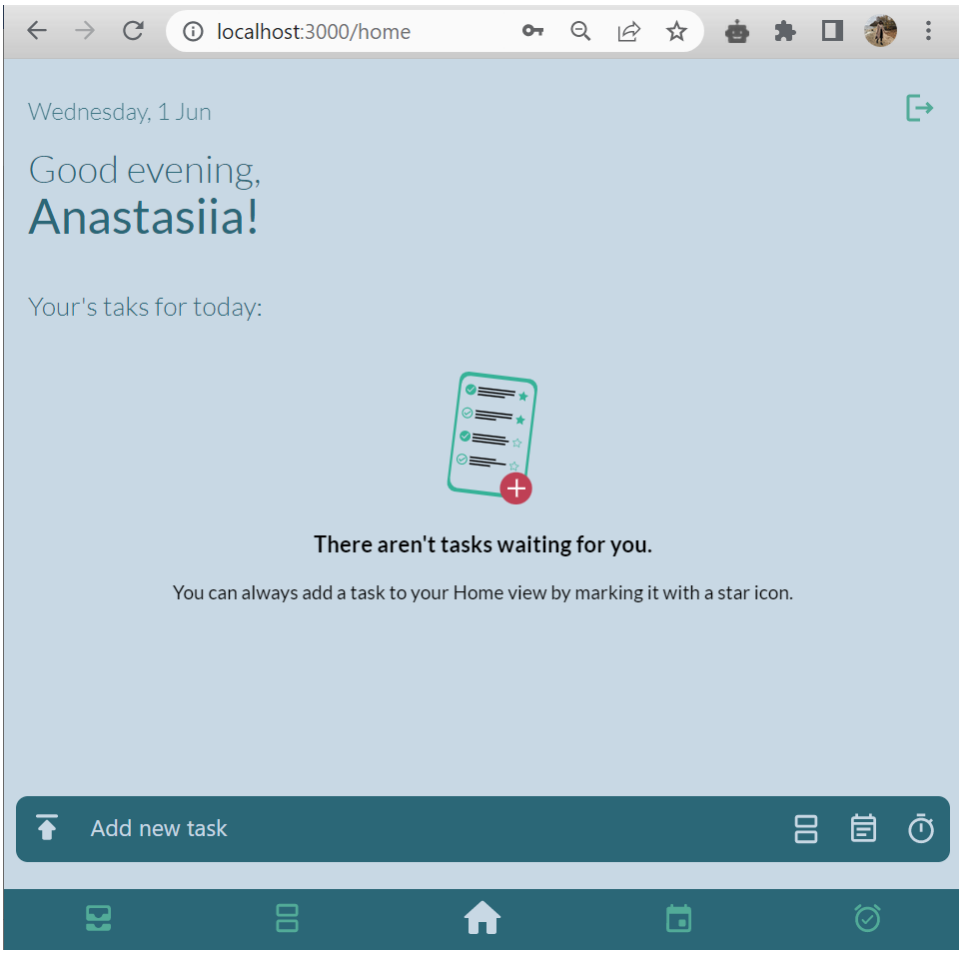

Рисунок 3.13 – Головний екран застосунку для авторизованого користувача

Так як користувач вперше зайшов у свій обліковий запис, в нього поки немає створених задач, однак внизу є панель *Add new task*, яка дозволяє досить швидко додати завдання.

Завдання можуть мати три необов'язкові атрибути, які додаються відповідними кнопками у правій частині панелі. Перша відповідає за проєкт, до якого відноситься задача. За відсутності проєктів зображене на рисунку 3.14 меню пропонує перейти на відповідну сторінку та створити проєкт, в інакшому випадку меню буде виглядати таким чином, як на рисунку 3.15.

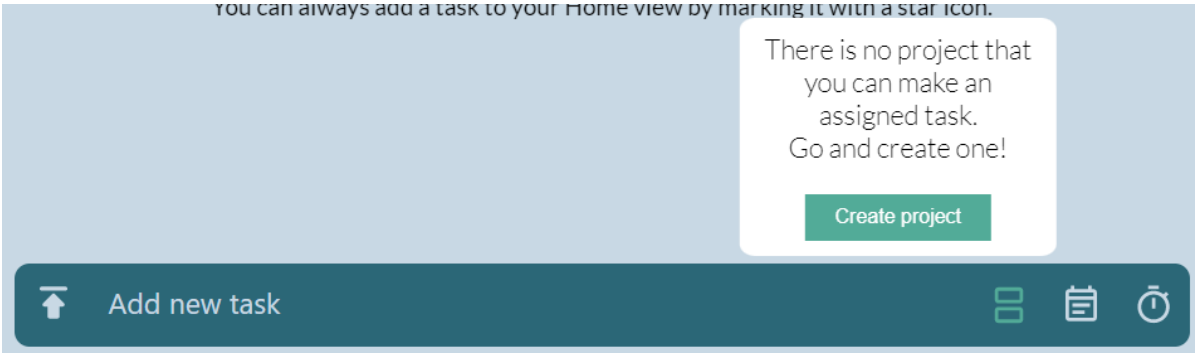

Рисунок 3.14 – Додавання задачі до проєкту за відсутності існуючих проєктів

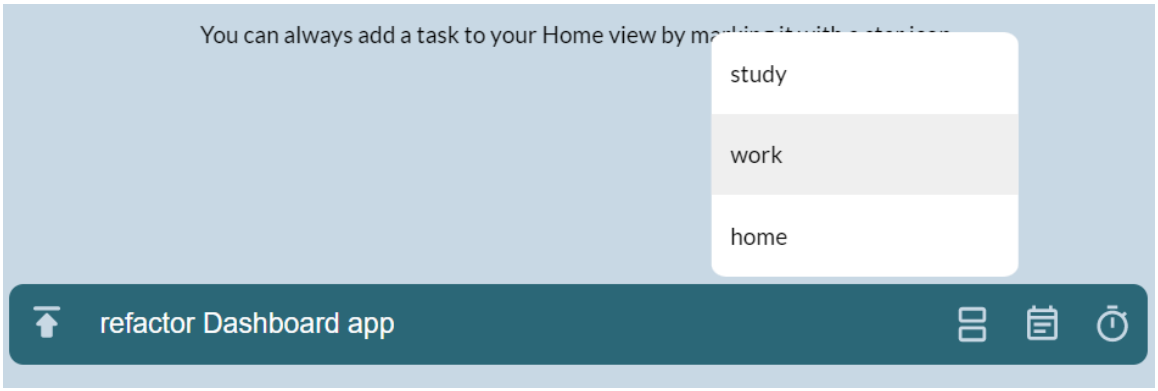

Рисунок 3.15 – Додавання задачі до існуючого проєкту

Другим атрибутом є дедлайн виконання задачі. Користувач може обрати або шаблон дати, або ж задати її самостійно (рис. 3.16).

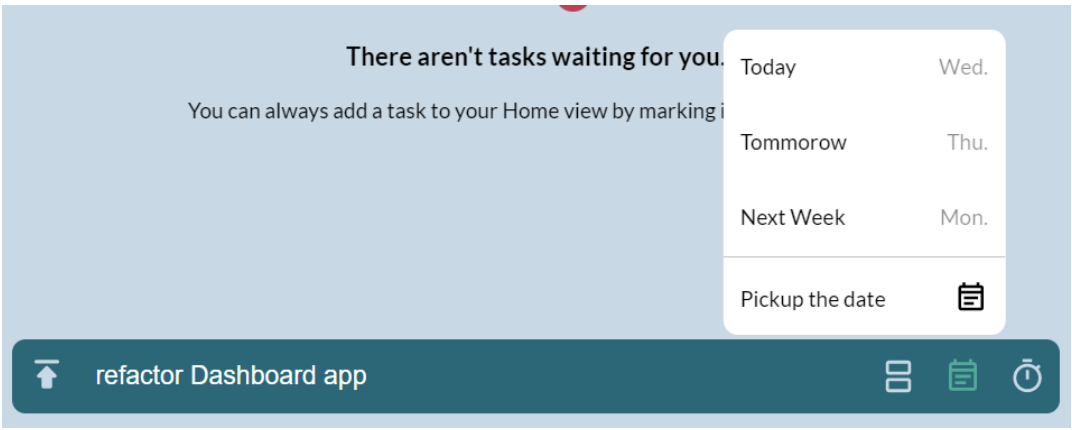

Рисунок 3.17 – Додавання дедлайну до задачі

Останнім атрибутом є додавання кількості «помідорів» для завершення даної задачі. Вона також може бути обрана або із шаблонів, або ж створена користувачем самостійно в залежності від його потреб (рис. 3.18).

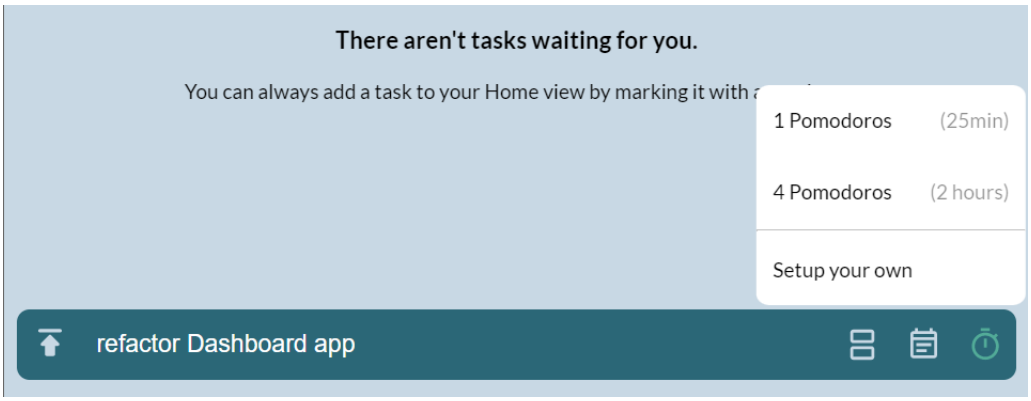

Рисунок 3.18 – Додавання «помідорів» до задачі

Після додавання всіх необхідних атрибутів задача з'являється на головному екрані (рис. 3.19), причому вона автоматично помічається як вибране. Якщо прибрати дану зірочку, то на домашній сторінці дане завдання відображатись не буде.

Рисунок 3.19 – Створена задача

Всі задачі можна побачити на сторінці списку задач (рис. 3.20). Причому вони одразу будуть розбиті на виконані та актуальні. За бажанням можна приховати всі минулі завдання. Також є можливість додавання задачі за допомогою нижньої панелі. У разі накопичення великої кількості карток із завданнями, вгорі присутній пошук за назвою.

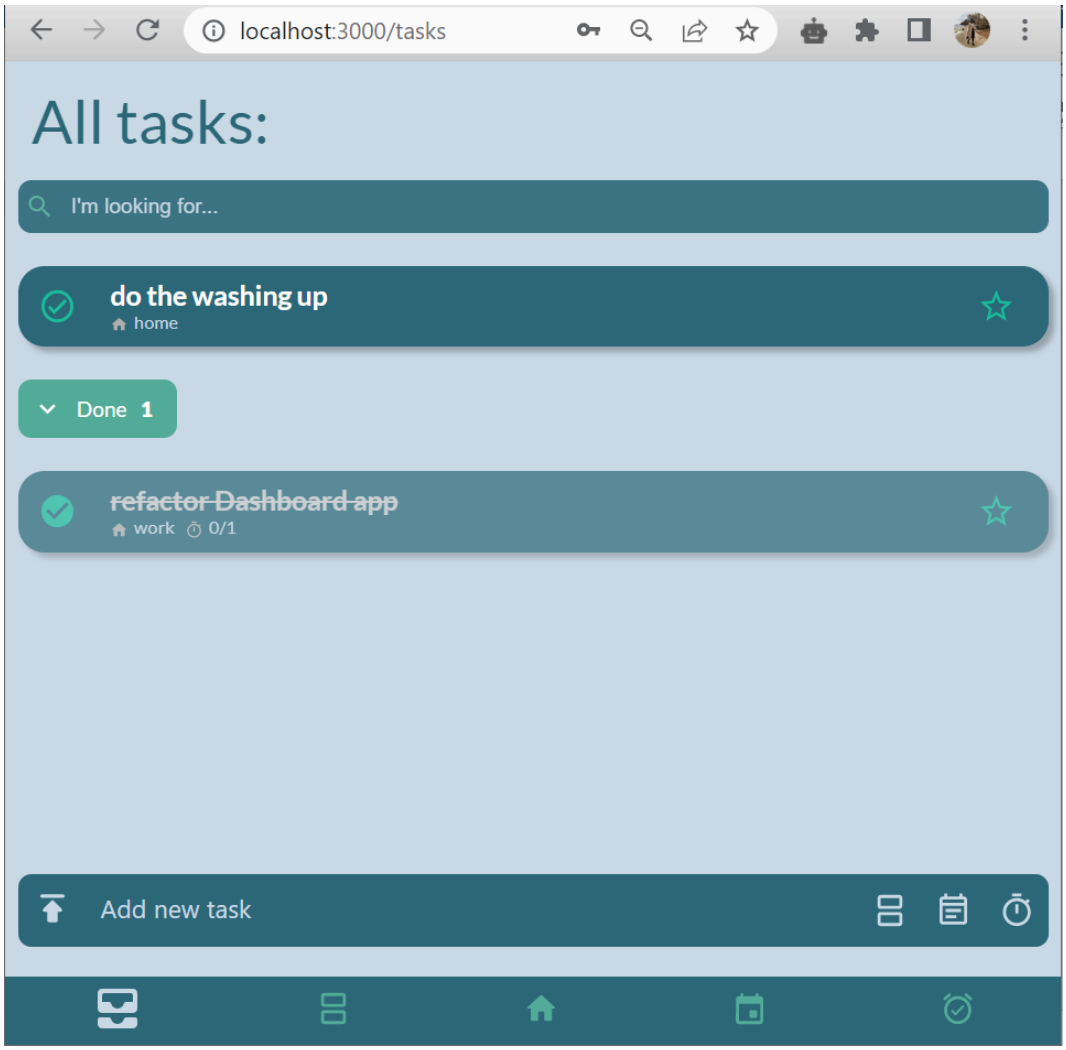

Рисунок 3.20 – Сторінка списку задач

Кожну із існуючих задач можна редагувати. Для цього необхідно клікнути на потрібну картку та у меню, яке з'явилося, змінити та/або додати необхідні атрибути (рис. 3.21).Причому примітка до задачі може містити не більше 150 символів. У полі *Add user* можна додати електрону адресу користувача, який зареєстрований у системі. Таким чином дана задача з'явиться і у нього, тобто користувачі матимуть спільний доступ. Також за допомогою іконки у правому нижньому кутку можна видалити дану задачу.

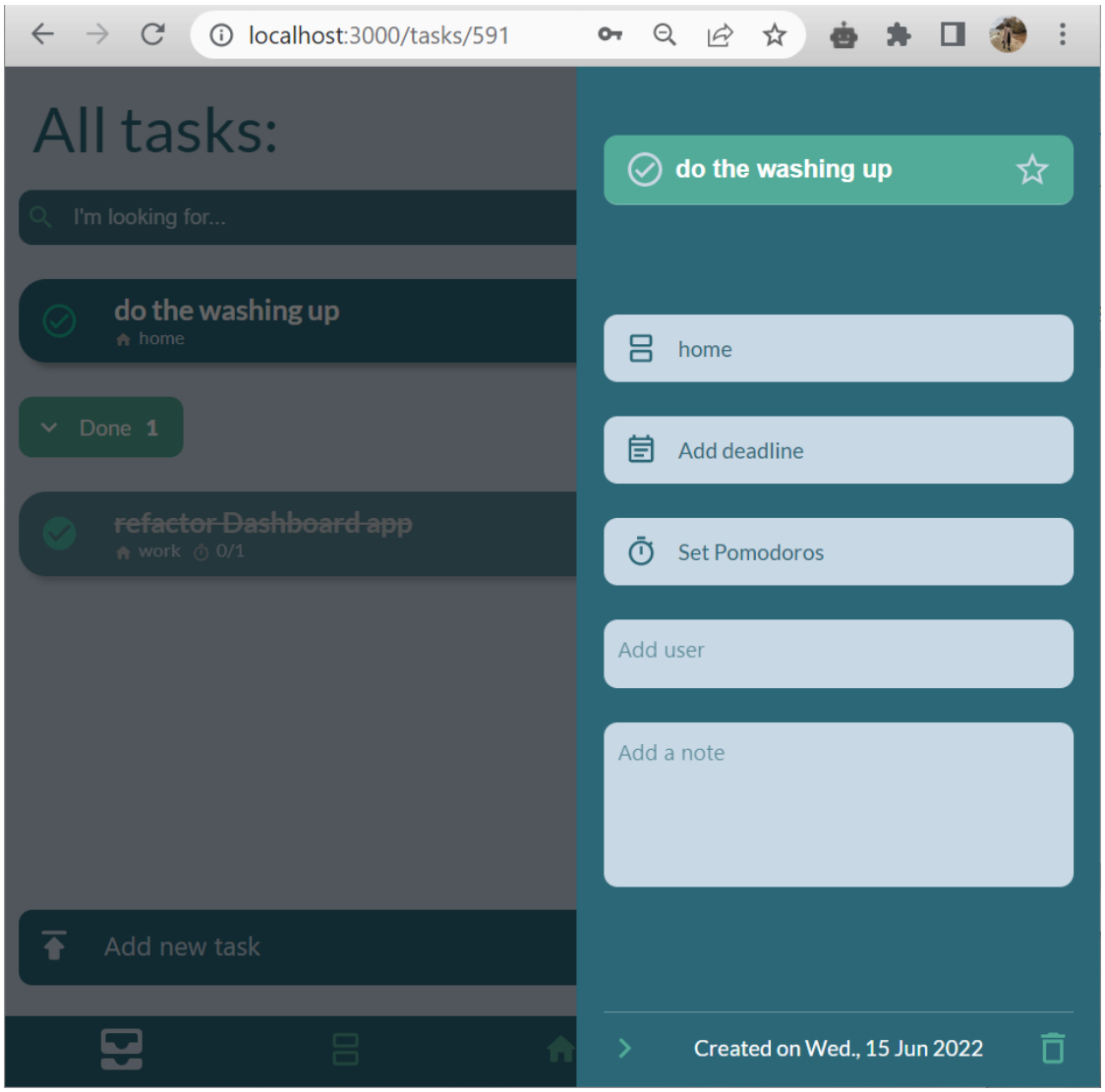

Рисунок 3.21 – Панель редагування задачі

Наступною сторінкою є сторінка проєктів, тобто свого роду контейнерів для задач (рис. 3.22). Картка кожного проєкту у лівому верхньому кутку відображає кількість задач, які він містить (рахуються як виконані, так і актуальні задачі). При створені ж кожен проєкт обов'язково має назву, а також додатково може містити опис та колір. Вже існуючий проєкт також можна як видаляти, так і редагувати.

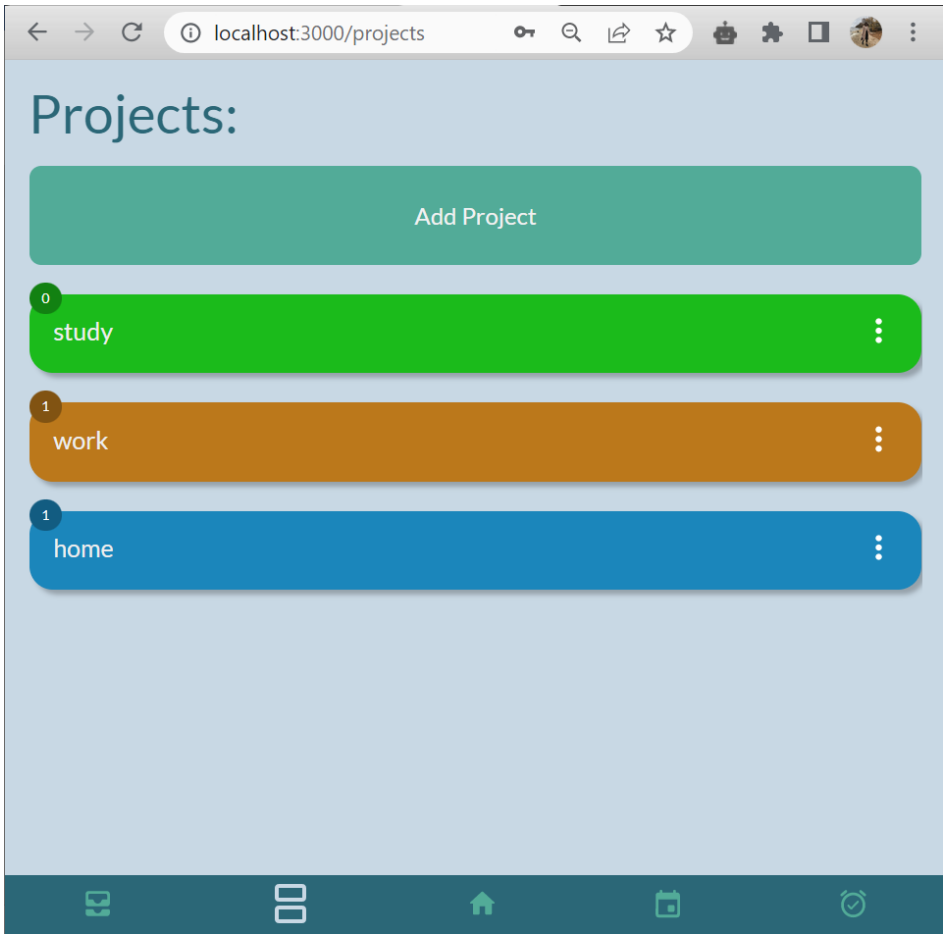

Рисунок 3.22 – Сторінка проєктів користувача

Важливою особливістю проєктів є те, що користувач може підлаштувати їх під себе. Одним із таких способів є створення пріоритетності задач. Тобто, можна створити чотири окремих проєкти із назвами P1, P2, P3, P4 та відповідно диференціювати їх за кольорами (зазвичай, червоний – P1, помаранчевий – P2, жовтий – P3 та зелений – P4). Таким чином, у перший проєкт можна закидати найбільш важливі задачі, у другий – із другим пріоритетом і т. д. Дана методика є досить дієвою і вона легко застосовується за допомогою тих засобів, що наявні у даному веб-застосунку.

Якщо користувач вирішує видалити проєкт, він побачить модальне вікно із підтвердженням даної дії (рис. 3.23). Однак важливо зазначити те, що задачі, які належать даному проєкту, видалені не будуть.

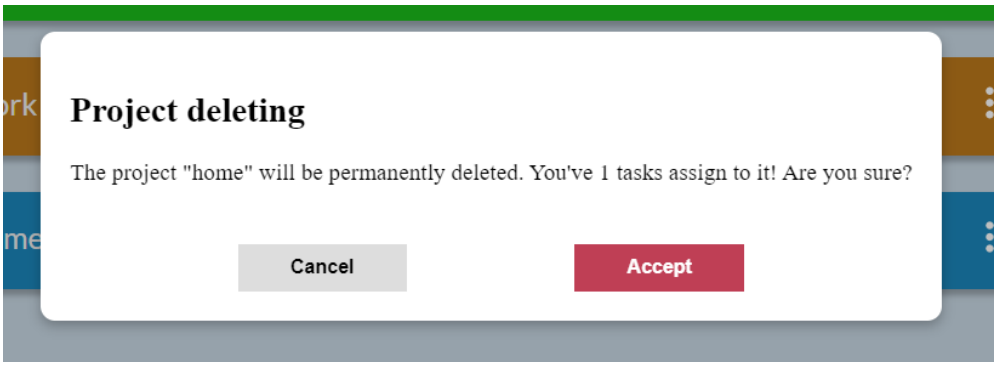

Рисунок 3.23 – Модальне вікно підтвердження видалення проєкту

Наступною сторінкою є календар (рис. 3.24), в якому користувач може передивлятися всі свої задачі по датам. Натиснувши на будь-яку дату, можна швидко створити задачу таким самим чином, як це робиться на інших сторінках. Крім того, створені задачі можна редагувати та видаляти за аналогією цієї функціональності на сторінці відображення списку задач.

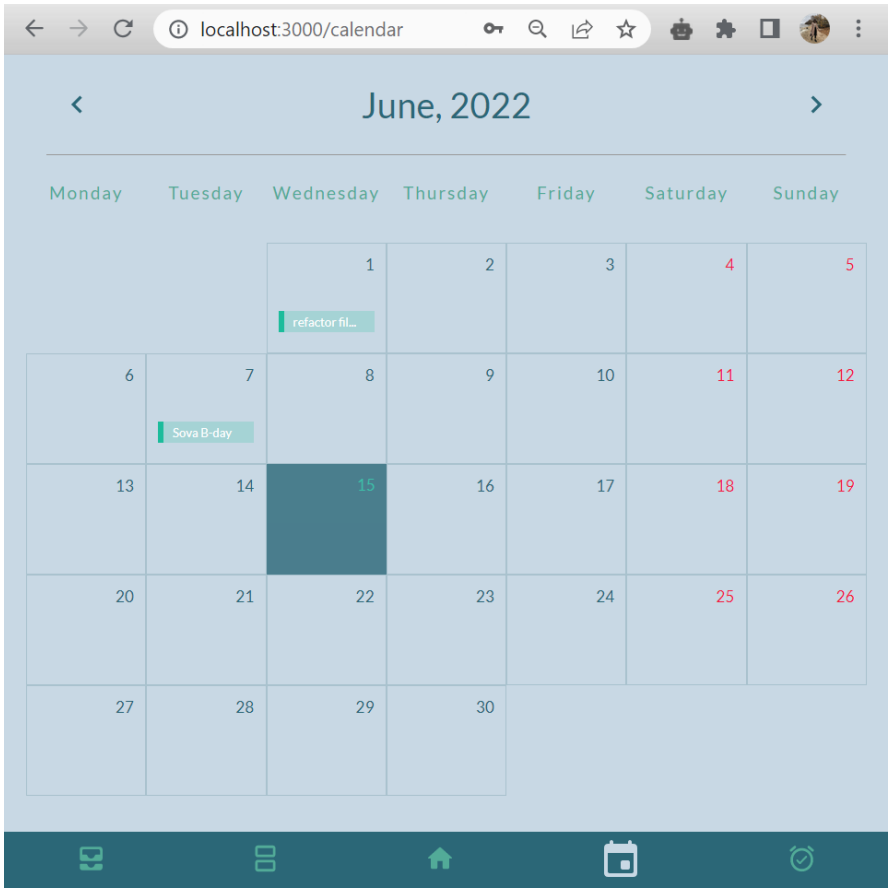

Рисунок 3.24 – Сторінка календаря

Останньою доступною сторінкою є сторінка таймера (рис. 3.25). Вона являє собою класичний вигляд таймера для застосування техніки помідора. За замовченням робота, коротка та довга перерви тривають 25 хвилин, 5 хвилин та 15 хвилин відповідно. Під самим годинником відображуються ті задачі, які потребують певну кількість «помідорів» для виконання.

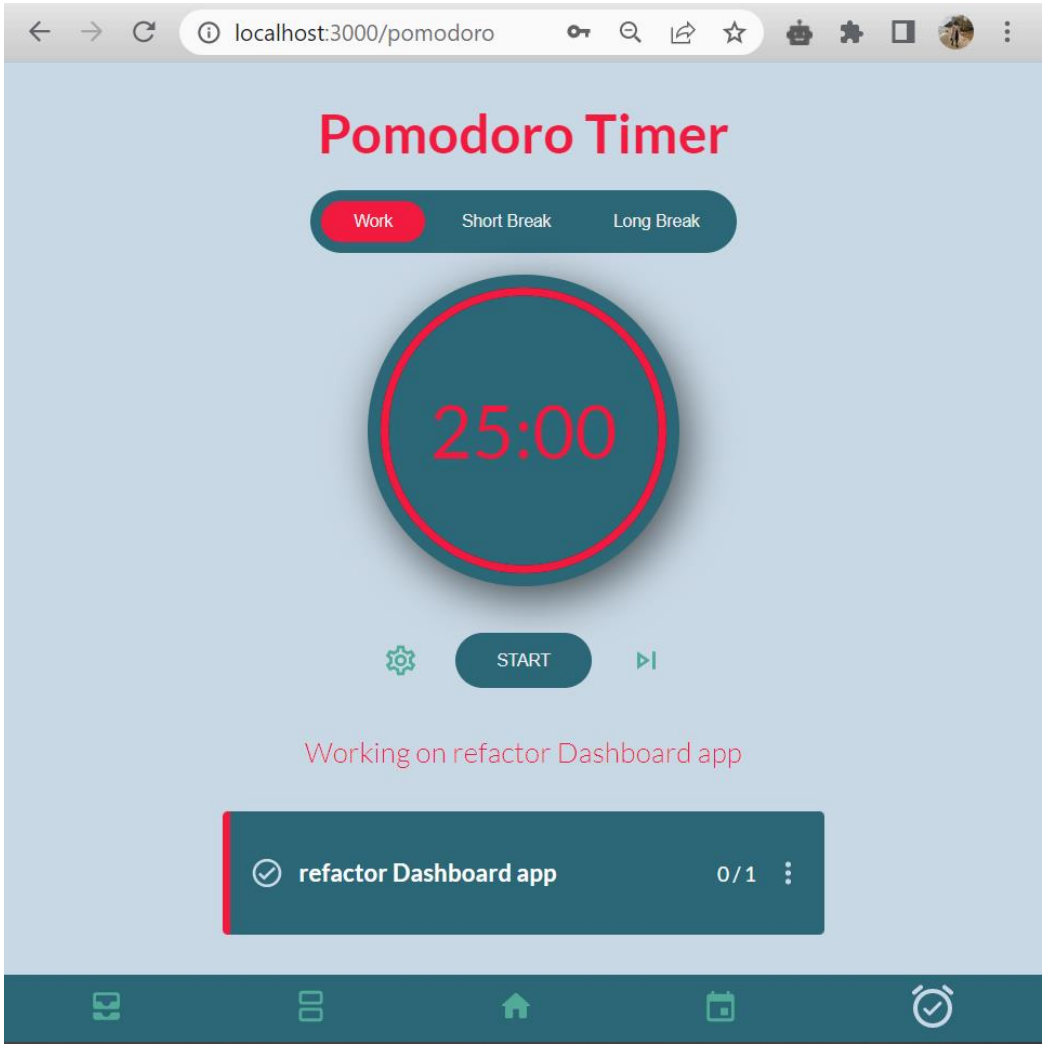

Рисунок 3.25 – Сторінка таймера

Для того, щоб запустити таймер, необхідно натиснути *Start*, а по закінченні часу користувач почує характерний звук. Також задача автоматично перейде у стан «виконана». Для того, щоб закінчити завчасно період роботи, можна натиснути на іконку переходу, яка знаходиться справа від кнопки *Start*.

Для персональної конфігурації таймера необхідно натиснути на іконку зліва від кнопки *Start* та у модальному вікні обрати необхідні налаштування (рис. 3.26).

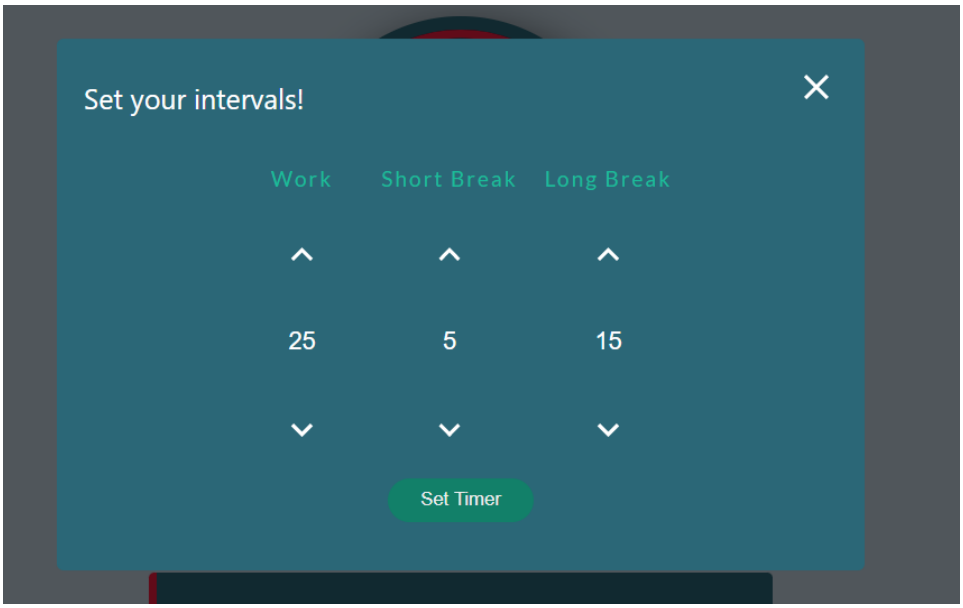

Рисунок 3.26 – Налаштування часових інтервалів таймера

Висновки до розділу 3

Отже, у даному розділі було розглянуто основні вимоги до створення веб-застосунку планування задач із підтримкою групових заходів. Було описано основні моменти програмної реалізації, а також продемонстровано основні функції готового застосунку.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Спеціальний розділ

ОХОРОНА ПРАЦІ

до кваліфікаційної роботи

на тему:

«ВЕБЗАСТОСУНОК ПЛАНУВАННЯ ЗАДАЧ ІЗ ПІДТРИМКОЮ ГРУПОВИХ ЗАХОДІВ»

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402.21810208

Виконала студентка 4-го курсу, групи 402
А. К. Гресс

«22» червня 2022 р.

Консультант к.т.н., доцент
А. О. Алексєєва

«22» червня 2022 р.

Миколаїв – 2022

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ВДТ	–	Візуальний дисплейний термінал
ЕОМ	–	Електронна обчислювальна машина
ПЕОМ	–	Персональні електронні обчислювальні машини

4. ОХОРОНА ПРАЦІ

Охорона праці при користуванні екранними пристроями

ВСТУП

У сучасному світі інформаційні технології відіграють надзвичайно важливу роль як у професійній діяльності людини, так і в її особистому житті. Якщо раніше в основному лише IT-компанії використовували електронні системи для підписання документів, зберігання, обміну та аналізу даних, то після розгортання пандемії COVID-19 велика кількість різноманітних установ та організацій впровадили електронний документообіг та почали більш активно користуватися новими технологіями. Безумовно, вплив впровадження інформаційних технологій у бізнес-процеси не тільки полегшує роботу працівників, але й підвищує ефективність робочих процесів, а також за певних умов робить інформацію більш захищеною. Однак у той же час потрібно усвідомлювати, що тривала робота із електронно-обчислювальними машинами викликає достатньо негативні зміні у функціонування організму, а отже однією із головних задач компаній є проведення профілактичних заходів щодо мінімізації шкідливого впливу комп’ютера на здоров’я користувача.

Проблеми, які виникають внаслідок довготривалої роботи перед монітором, можуть бути пов’язані не лише із фізичним здоров’ям, але і з психічним. Найбільш поширеними є розлади зору, головний біль, дратівливість, напруга та стрес.

Важливо зазначити, що робота за екранними пристроями вимагає довгого знаходження у сидячому положенні, що також нерідко є причиною погіршення стану здоров’я людини, а саме болі у спині, послаблення м’язів, загострення варикозної хвороби.

4.1 Нормативна база в галузі охорони праці при роботі з екранними пристроями

Екранні пристрої – електронні засоби для відтворення будь-якої графічної або алфавітно-цифрової інформації (на основі електронно-променевої трубки, рідкокристалічні, плазмові, проекційні, органічні світлодіодні монітори та інші новітні розробки у сфері інформаційних технологій) [20].

Мінімальні вимоги безпеки та захисту здоров’я під час роботи з екранними пристроями всіх типів і моделей установлюють Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями, затверджені наказом Міністерства соціальної політики від 14.02.2018 № 207, який поширюється на всіх суб’єктів господарювання незалежно від форм власності, організаційно-правової форми і видів діяльності та встановлює мінімальні вимоги безпеки та захисту здоров’я під час здійснення роботи, пов’язаної з використанням екранних пристроїв незалежно від їхнього типу та моделі [20].

Ще одним нормативно-правовим актом, яким має керуватися роботодавець з метою створення належних умов праці при роботі з екранними пристроями є Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, затвердженими постановою Головного державного санітарного лікаря України 10.12.1998 р. № 7, що застосовуються на підприємствах, організаціях, установах незалежно від форми власності та поширюються на умови й організацію праці при роботі з візуальними дисплейними терміналами усіх типів вітчизняного та закордонного виробництва на основі електронно-променевих трубок, якими укомплектовані електронно-обчислювальні машини колективного використання та персональні ЕОМ [21].

4.2. Загальні вимоги до роботодавців

Задля забезпечення безпеки та захисту здоров’я під час здійснення роботи, пов’язаної з використанням екранних пристроїв роботодавець повинен виконувати дотримуватися певних вимог. Однак за певних умов, а саме – якщо це не суперечить чинному законодавству, то роботодавець має право встановлювати більш жорсткі та/або спеціальні вимоги безпеки і захисту здоров’я та життя працівників під час роботи з екранними пристроями. Головне, що потрібно пам’ятати під час облаштування робочого місця працівника з екранними пристроями необхідно обирати таке устаткування, яке не створює зайвого шуму та не виділяє надлишкового тепла.

Нижче наведено частину вимог з документу про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями:

– роботодавець повинен поінформувати працівників про умови праці та наявність на їх робочих місцях небезпечних та шкідливих виробничих факторів;

– роботодавець повинен вжити відповідних заходів, щоб забезпечити відповідність робочого місця працівника;

– під час облаштування робочого місця працівника з екранними пристроями необхідно обирати таке устаткування, яке не створює зайвого шуму та не виділяє надлишкового тепла;

– роботодавець повинен за рахунок тривалості робочої зміни організувати внутрішні регламентовані перерви для відпочинку відповідно до ДСанПІН 3.3.2.007-98 [22];

– роботодавець зобов’язаний за необхідності проводити лабораторні дослідження умов праці працівників з метою виявлення шкідливих і небезпечних факторів виробничого середовища, важкості та напруженості трудового процесу.

4.3 Вимоги до приміщення з використанням екранних пристроїв

Приміщення, у яких робітники використовують електронно-обчислювальні машини, повинні відповідати певному ряду вимог, які перевіряються спеціальними службами, призначеними для нагляду за дотриманням вимог охорони праці на підприємствах.

Нижче наведено частину вимог з Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин:

– розміщення робочих місць з ВДТ ЕОМ і ПЕОМ у підвальних приміщеннях, на цокольних поверхах заборонено;

– площа на одне робоче місце має становити не менше ніж 6,0 м^2, а об'єм не менше ніж 20,0 м^3;

– приміщення для роботи з ВДТ повинні мати природне та штучне освітлення відповідно до СНиП II-4-79 [23];

– природне освітлення має здійснюватись через світлові прорізи, орієнтовані переважно на північ чи північний схід і зебезпечувати коефіцієнт природною освітленості не нижче ніж 1,5%. Розраховується цей коефіцієнт за методикою, викладеною в СНиП II-4-79 [23];

– у приміщеннях з ЕОМ слід щоденно робити вологе прибирання;

– приміщення з ЕОМ мають бути оснащені аптечками першої медичної допомоги;

– при приміщеннях з ВДТ мають бути обладнані побутові приміщення для відпочинку під час роботи, кімната психологічного розвантаження. В кімнаті психологічного розвантаження слід передбачити встановлення пристроїв для приготування й роздачі тонізуючих напоїв, а також місця для занять фізичною культурою (СНиП 2.09.04.-87 [24]);

Нижче наведено частину вимог з документу наказу про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями від 14.02.2018:

– робочі місця працівників з екранними пристроями мають бути спроектовані так і мати такі розміри, щоб працівники мали простір для зміни робочого положення та рухів;

– усе випромінювання від екранних пристроїв має бути зведене до гранично допустимого рівня;

– освітлення робочого місця працівника з екранними пристроями має створювати відповідний контраст між екраном і навколишнім середовищем (з урахуванням виду роботи) та відповідати вимогам ДСанПІН 3.3.2.007-98 [22];

– мікроклімат виробничих приміщень з робочими місцями працівників з екранними пристроями має підтримуватись на постійному рівні та відповідати вимогам Санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042-99 [25];

– робочий стіл або робоча поверхня повинні бути достатнього розміру та мати поверхню з низькою відбивною здатністю, допускати гнучкість під час розміщення екрана, клавіатури, документів і відповідного устаткування;

– робоче крісло має бути стійким і дозволяти працівнику з екранними пристроями легко рухатися та займати зручне положення. Сидіння має регулюватися по висоті, спинка сидіння – як по висоті, так і по нахилу. Слід передбачати підніжку для тих, кому це необхідно для зручності.

Крім цього для зменшення впливу негативних факторів на працівника, виробниче приміщення повинно відповідати нормам щодо параметрів мікроклімату, освітлення, шуму та вібрації, рівні електромагнітного та іонізуючого випромінювання.

По-перше, на усіх робочих місцях із електронно-обчислювальними машинами обов’язково має бути забезпечено оптимальні значення параметрів

температури відносної вологості й рухливості повітря (ГОСТ 12.1.005-88 [26], СН 4088-86 [27]) (табл. 3.1).

Таблиця 4.1 – Нормовані величини температури, відносної вологості та швидкості руху в робочій зоні виробничих приміщень з ВДТ

Пора року	Категорія робіт	Оптимальна температура повітря, °С, не більше	Оптимальна відносна вологість повітря, %	Оптимальна швидкість руху повітря, м/с
Холодна	Легка – Іа	22-24	40-60	0,1
	Легка – Іб	21-23	40-60	0,1
Тепла	Легка – Іа	23-25	40-60	0,1
	Легка – Іб	22-24	40-60	0,2

До категорії робіт Іа належать, що виконується робота сидячі і не потребує фізичного напруження людини, до категорії робіт Іб належать, що робота виконується сидячі, стоячи або пов’язані з ходінням та потребує деякого фізичного напруження.

По-друге, показники рівню позитивних та негативних іонів в повітрі приміщень з ВДТ мають відповідати санітарно – гігієнічним нормам №2152-80 [28] (табл. 3.2).

Таблиця 4.2 – Рівні іонізації повітря приміщень при роботі на ВДТ

Рівні	Кількість іонів в 1 $см^3$ повітря	
	n+	n-
Мінімально необхідний	400	600
Оптимальний	1500-3000	3000-5000
Максимально допустимий	50000	50000

Вимірювання кількості іонів та його полярності порядку поточного нагляду проводиться 1 разів у квартал. Вимірювання проводяться також у випадках:

– встановлення нових або відремонтованих іонізаторів;

– організації нових робочих місць;

– впровадження нових технологічних процесів, що потенційно можуть змінити іонний режим у зоні дихання персоналу.

Що до штучного освітлення в приміщеннях з робочими місцями, обладнаними ВДТ ЕОМ та ПЕОМ, то воно має здійснюватись системою загального івномірного освітлення. Зазначення освітлення освітленості на поверхні робочого столу в зоні розміщення документів має становити 300-500 лк. Якщо ці значення освітленості неможливо забезпечити системою загального освітлення, допускається використовувати місцева освітлення. При цьому світильники місцевого освітлення слід встановлювати таким чином, щоб не створювати бліків на поверхні екрана, а освітленість екрана має не перевищувати 300 лк.

У якості джерела світла у приміщеннях для штучного освітлення мають використовуватися люмінесцентні лампи типу ЛБ. При використання відбитого освітлення у виробничих та адміністративних приміщеннях допускається використання метало-галогенних ламп, які мають потужність 250Вт.

Допускається використання ламп розжарювання у світильниках місцевого освітлення.

Інтенсивність потоків інфрачервоного випромінювання має не перевищувати допустимих значень, які зазначені у ДСН 3.3.6.042-99 [25].

Інтенсивність потоків ультрафіолетового випромінювання має не перевищувати допустимих значень, які зазначені у СН 4557-88 [29].

Потужність експозиційної дози рентгенівського випромінювання на відстані 0,05м від екрану та корпусу комп’ютера не повинна перевищувати 0,1мбер/рік (100мкР/рік).

4.4 Вимоги до роботи з екранними пристроями

В обов’язковому порядку працівники, основним місцем роботи яких є місце за екранними пристроями, повинні у свою чергу також дотримуватись певних вимог. Нижче наведено частину вимог з наказу Міністерством соціальної політики України «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями»

На сьогодні існує ряд вимог, які повинні використовувати працівники на підприємствах, основним місцем роботи яких є місце за персональним комп’ютером. Нижче наведено частину вимог з наказу Міністерства соціальної політики щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями від 14.02.2018 [20].

Мінімальні вимоги безпеки під час роботи з екранними пристроями:

– щодня перед початком роботи необхідно очищати екранні пристрої від пилу та інших забруднень;

– після закінчення роботи екранні пристрої слід відключати від електричної мережі;

– у разі виникнення аварійної ситуації необхідно негайно відключити екранний пристрій від електричної мережі.

Не допускається:

– виконувати технічне обслуговування, ремонт і налагодження екранних пристроїв безпосередньо на робочому місці працівника під час роботи з екранними пристроями;

– відключати захисні пристрої, самочинно проводити зміни у конструкції та складі екранних пристроїв або їх технічне налагодження;

– працювати з екранними пристроями, у яких під час роботи виникають нехарактерні сигнали, нестабільне зображення на екрані та інші несправності.

Мінімальні вимоги безпеки до екранних пристроїв:

– екранні пристрої не мають бути джерелом ризику для працівників;

– усе випромінювання, за винятком видимої частини електромагнітного спектра, має бути зведене до незначного рівня з погляду безпеки і охорони здоров’я працівників;

– символи на екранних пристроях мають бути чіткими, відповідного розміру, між символами і рядками символів має бути належна відстань;

– зображення на екрані має бути стабільним, без миготінь або інших видів нестабільності;

– яскравість та/або контрастність символів має легко регулюватися працівником під час роботи з екранними пристроями, а також швидко адаптуватися до навколишніх умов;

– вибираючи екрани, слід надавати перевагу таким екранам, які легко та вільно повертаються і нахиляються відповідно до потреби працівника;

– за необхідності може використовуватись окрема підставка або регульований стіл для розміщення екрана;

– екран не має відблискувати або відбивати світло, щоб не викликати дискомфорту у працівника під час роботи з екранними пристроями.

– обираючи клавіатуру, слід надавати перевагу такій, яка відкидається і є відокремленою від екрана, щоб працівник міг вибрати зручну робочу позу й уникнути втоми рук;

– поверхня клавіатури має бути матовою, щоб уникнути віддзеркалювання. Розташування клавіш і самі клавіші мають полегшувати роботу із клавіатурою. Позначення клавіш повинно бути достатньо контрастним і розбірливим;

– устаткування, яке входить до робочої станції, не має виділяти надлишкового тепла, яке може спричинити незручності працівникам під час роботи з екранними пристроями.

– під час розробки, вибору, замовлення та модифікації програмного забезпечення, а також під час розробки завдань, що передбачають використання устаткування з екранними пристроями, роботодавець має керуватися таким програмним забезпеченням, яке відповідає розв'язуваним завданням і є простим у використанні, а де необхідно – адаптованим до рівня знань і досвіду працівника.

4.5. Вимоги щодо режиму відпочинку та праці на підприємствах з екранними пристроями

Не менш важливим аспектом при організації роботи, яка пов'язана із електронно-обчислювальними машинами, є внутрішньо змінні регламентовані перерви на відпочинок. Вони повинні містити додаткові короткочасні перерви, які запобігають появі ознак стомлення та зниження продуктивності працівників.

При виконанні протягом дня робіт, що належать до різних видів трудової діяльності, за основну роботу з ВДТ ЕОМ і ПЕОМ слід вважати таку, що займає не менше 50% часу впродовж робочої зміни мають передбачатися [21]:

– перерви для відпочинку і вживання їжі (обідні перерви)

– перерви для відпочинку і особистих потреб (згідно з трудовими нормами);

– додаткові перерви, що вводяться для окремих професій з урахуванням особливостей трудової діяльності.

Встановлюються такі внутрішньозмінні режими праці та відпочинку при роботі з ЕОМ при 8-годинній денній робочій зміні в залежності від характеру праці:

– для розробників програм із застосуванням ЕОМ, слід призначати регламентовану перерву для відпочинку тривалістю 15 хвилин через кожну годину роботи за ВДТ;

– для операторів із застосування ЕОМ, слід призначати регламентовані перерви для відпочинку тривалістю 15 хвилин через кожні дві години;

– для операторів комп'ютерного набора слід призначати регламентовані перерви для відпочинку тривалістю 10 хвилин після кожною години роботи за ВДТ.

У всіх випадках, коли виробничі обставини не дозволяють застосувати регламентовані перерви, тривалість безперервної роботи з ВДТ не повинна перевищувати 4 години.

При 12-годинній робочій зміні регламентовані перерви повинні встановлюватися в перші 8 годин роботи аналогічно перервам при 8-годинній робочій зміні, а протягом останніх 4-х годин роботи, незалежно від характеру трудової діяльності, через кожну годину тривалістю 15 хвилин.

Також із метою зменшення негативного впливу монотонності є доцільним застосовувати чергування операцій усвідомленого тексту і числових даних (зміна змісту роботи). Чередування вводу даних та редагування текстів.

Активний відпочинок має полягати у виконанні комплексу гімнастичних вправ, спрямованих на зняття нервового напруження, м'язове розслаблення, відновлення функцій фізіологічних систем, що порушуються протягом трудового процесу, зняття втоми очей, поліпшення мозкового кровообігу і працездатності

За умови високого рівня напруженості робіт з ВДТ показане психологічне розвантаження у спеціально обладнаних приміщеннях (в кімнатах психологічного розвантаження) під час регламентованих перерв або в кінці робочого дня.

ВИСНОВКИ ДО РОЗДІЛУ 4

В результаті виконання спеціальної частини з охорони праці було досліджено умови організації роботи працівників на підприємствах та установах, а також правила роботи при користуванні електронно-обчислювальними пристроями.

Так як у сучасному світі неможливо уявити ведення бізнес-процесів без комп’ютерів, правила та вимоги, прописані у нормативних документах є надзвичайно актуальними. Однак варто пам’ятати, що довготривала робота за екранними пристроями чревате негативними наслідками для здоров’я людини, адже йде навантаження на зір, опорно-руховий апарат та на нервову систему.

Основними умовами, які впливають на організм людини є такі фактори, як мікроклімат (температура, вологість, швидкість руху повітря), освітлення, інфрачервоне випромінювання, шум, вібрація та інші умови.

Дотримання нормативів та вимог щодо охорони праці під час трудової діяльності як зі сторони роботодавця, так і самими працівниками дозволяють мінімізувати шкідливий вплив на здоров’я людини.

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи було проведено аналіз предметної сфери планування задач, який допоміг визначити основні критерії для створення ефективної системи тайм-менеджменту. Також було розглянуто аналоги вже існуючих додатків для керування часом. Перед безпосереднім створенням веб-застосунку було визначено інструментальні засоби розробки шляхом розглядання переваг та недоліків таких технологій, як React, Redux, Firebase.

Після цього на основі визначених вимог було створено веб-застосунок за допомогою бібліотеки React, який дозволяє оптимізувати процес персонального тайм-менеджменту, а також має підтримку групових заходів. Функціональність системи дозволяє користувачу групувати задачі за проєктами, відслідковувати виконані та невиконані завдання, переглядати календар із дедлайнами, а також створювати заходи для декількох користувачів.

Особливістю застосунку є можливість застосування практики помідору до виконання своїх задач. Це є досить ефективним методом для організації самодисципліни та підвищення персональної продуктивності.

СПИСОК ВИКОРИСТАННИХ ДЖЕРЕЛ

1. Time Management Statistics & Facts (New 2021 Research): веб-сайт. URL: https://development-academy.co.uk/news-tips/time-management-statistics-2021-research/ (дата звернення: 21.05.2022).

2. Аллен Д. Як упорядкувати справи. Мистецтво продуктивності без стресу / Девід Аллен; Пер. з анг. К. Козачук, Н. Кузьменко. – К. : Вид. група КМ-БУКС, 2018. 392 с.

3. Кох Р. Принцип 80/20. Секрет досягнення більшого за менших витрат . Річард Кох; Пер. з англ. Н.Лавської, О. Лобастової – К. : Вид. група КМ-БУКС, 2019. 400 с.

4. Cirillo F. The Pomodoro Technique. The Life-Changing Time-Management System / Francesco Cirillo – Ebury Press, 2018. 130 p.

5. Address at the Second Assembly of the World Council of Churches, Evanston, Illinois : веб-сайт. URL: https://www.presidency.ucsb.edu/documents/address-the-second-assembly-the-world-council-churches-evanston-illinois (дата звернення: 26.05.2022).

6. Covey St. The 7 Habits of Highly Effective People: Powerful Lessons in Personal Change / Stephen R. Covey – Free Press, 2004. 384 p.

7. Task Management Software Market by Business Function (Marketing, Human Resource, Finance), Component (Software and Services), Deployment Type (Cloud and On-Premises), Organization Size, Industry Vertical, and Region - Global Forecast to 2023: веб-сайт. URL: https://www.marketsandmarkets.com/Market-Reports/task-management-software-market-171016683.html (дата звернення: 27.05.2022).

8. Trust Radius: веб-сайт. URL: https://www.trustradius.com/ (дата звернення: 27.05.2022).

9. 2021 Developer Survey: веб-сайт. URL: https://insights.stackoverflow.com/survey/2021#most-popular-technologies-language-prof (дата звернення: 28.05.2022).

10. Websites using React: веб-сайт. URL: https://trends.builtwith.com/websitelist/React (дата звернення: 28.05.2022).

11. Redux Documentation: веб-сайт. URL: https://react-redux.js.org/introduction/getting-started (дата звернення: 21.04.2022).

12. Redux Toolkit Documentation: веб-сайт. URL: https://redux-toolkit.js.org/ (дата звернення: 21.04.2022).

13. Comparing database types: how database types evolved to meet different needs: веб-сайт. URL: https://www.prisma.io/dataguide/intro/comparing-database-types (дата звернення: 21.04.2022).

14. React JS Documentation: Create a New React App): веб-сайт. URL: https://reactjs.org/docs/create-a-new-react-app.html (дата звернення: 21.04.2022).

15. JavaScript Patterns: Build Better Applications with Coding and Design Patterns / Stoyan Stefanov – O'Reilly Media, 2010 – 236 pp.

16. Learning React: Modern Patterns for Developing React Apps / Eve Porcello – O'Reilly Media, 2020 – 310 pp.

17. React Explained: Your Step-by-Step Guide to React / Zac Gordon – Independently published, 2020. – 366 pp.

18. Single Page Web Applications: JavaScript end-to-end / Josh Powell – Manning Publications, 2013 – 432 pp.

19. SPA Design and Architecture: Understanding Single Page Web Applications / Emmit Scott – Manning Publications, 2015 – 275 pp.

20. Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями URL: http://zakon3.rada.gov.ua/laws/show/z0508-18 (дата звернення: 20.05.2022).

21. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН

3.3.2.007-98 URL: https://zakon.rada.gov.ua/rada/show/v0007282-98#Text (дата звернення: 20.05.2022).

22. Державні санітарні правила і норми роботи з ВДТ ЕОМ ДСанПІН 3.3.2.007-98 URL: http://mozdocs.kiev.ua/view.php?id=2445 (дата звернення: 20.05.2022).

23. СНиП II-4-79. Природне і штучне освітлення URL: https://dnaop.com/html/45036/doc- СНиП_II-4-79_ (дата звернення: 20.05.2022).

24. СНиП 2.09.04.-87. Адміністративні і побутові будівлі URL: https://dnaop.com/html/54074/doc -СНиП_2.09.04-87_ (дата звернення: 20.05.2022).

25. ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень URL: https://dnaop.com/html/34094/doc -ДСН_3.3.6.042-99 (дата звернення: 22.05.2022).

26. ГОСТ 12.1.005-88.ССБП URL: http://docs.cntd.ru/document/1200003608 (дата звернення: 22.05.2022).

27. СН 4088-86. Санітарні норми мікроклімату виробничих приміщень URL: http://docs.cntd.ru/document/901710059 (дата звернення: 22.05.2022).

28. Санітарно – гігієнічним нормам №2152-80 URL: https://dnaop.com/html/2296/doc -ГН_2152-80 (дата звернення: 22.05.2022).

29. СН 4557-88. Санітарні норми ультрафіолетового випромінювання URL: https://dnaop.com/html/2299/doc -СН_4557-88 (дата звернення: 22.05.2022).

ДОДАТОК А

Код методів реєстрації та авторизації користувачів

```
const handleLogin = async (e) => {
  e.preventDefault();
  try {
    await app
      .auth()
      .signInWithEmailAndPassword(
        values.email, values.password,
      )
      .then((userCredential) => userApi.getUser(userCredential.user.uid))
      .then((user) => {
        localStorage.setItem(USER_TOKEN, user[0].userId);
        dispatch(setUser({
          email: user[0].email,
          id: user[0].userId,
          nickname: user[0].name,
        }));
        push('/home');
      });
  } catch (error) {
    setErrorMessage(WRONG_LOGIN_MESSAGE);
  }
};

const handleRegistration = async (e) => {
  e.preventDefault();
  if (!validate()) return;
```

```
  try {
    await app
      .auth()
      .createUserWithEmailAndPassword(values.email, values.password)
      .then((userCredential) => {
        userApi.createUser({
          userId: userCredential.user.uid,
          name: values.name,
          email: values.email,
          token: generateToken(64),
        });
        push('/');
      });
  } catch (error) {
    setIsAlertShown(true);
    setAlertMessage(error.message);
  }
};
```

ДОДАТОК Б

Service API для сутності User

```
import { API } from '../serviceApi';

const USER_API = (API) => {
  const authUser = async (userData) => {
    const { data } = await API.api.post('users.json', userData);

    localStorage.setItem('USER_TOKEN', data.name);
    return data;
  };

  const createUser = async (userData) => {
    const { data } = await API.api.post('users.json', userData);
    return data;
  };

  const getUsers = async () => {
    let { data } = await API.api.get('users.json');
    if (!data) return [];
    data = Object.entries(data);
    const items = data.map((item) => ({ ...item[1], id: item[0] }));
    return items;
  };

  const getUser = async (userId) => {
    let { data } = await API.api.get('users.json');
    if (!data) return [];
```

```
    data = Object.entries(data);
    const items = data
      .filter((item) => item[1].userId === userId)
      .map((item) => ({ ...item[1], id: item[0] }));
    return items;
  };

  return {
    authUser,
    createUser,
    getUsers,
    getUser,
  };
};

export const userApi = USER_API(API);
```

ДОДАТОК В

Redux Slice для сутності User

```
import { createSlice } from '@reduxjs/toolkit';

const userSlice = createSlice({
  name: 'user',
  initialState: {
    nickname: null,
    email: null,
    password: null,
    token: null,
    id: null,
  },
  reducers: {
    setUser: (state, action) => {
      const {
        email, password, nickname, id, token,
      } = action.payload;
      state.email = email;
      state.password = password;
      state.nickname = nickname;
      state.id = id;
      state.token = token;
    },
    removeUser: (state) => {
      state.email = null;
      state.password = null;
      state.nickname = null;
```

```
      state.id = null;
      state.token = null;
    },
  },
});

export const { setUser, removeUser } = userSlice.actions;

export default userSlice.reducer;
```