Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра інформаційних систем та технологій
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня бакалавра

студента Морозова Богдана Денисовича
(ПІБ)
академічної групи 123-16-1
(шифр)
спеціальності 123 Комп'ютерна інженерія
(код і назва спеціальності)

за освітньо-професійною програмою 123 Комп'ютерна інженерія
(офіційна назва)
на тему «Комп'ютерна система управління персоналом підприємства «ModernDesign» з детальним опрацюванням користувацького інтерфейсу та побудови і налаштування корпоративної мережі»
(назва за наказом ректора)

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи	проф. Гнатушенко В.В.			
розділів:				
апаратний розділ	доц. Ткаченко С.М.			
розробка застосунку	проф. Гнатушенко В.В.			
економічний розділ	ст. викл. Яремчук І.О.			
охорона праці	доц. Іконніков М.Ю.			

Рецензент				

Нормоконтролер	проф. Цвіркун Л.І.			

Дніпро
2020

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних систем
та технологій
(повна назва)
______ проф. Гнатушенко В.В.
(підпис) (прізвище, ініціали)
____

ЗАВДАННЯна кваліфікаційну роботу
ступеня бакалавр

студента Морозова Б.Д. **академічної групи** 123-16-1
(прізвище та ініціали) (шифр)
спеціальності 123 «Комп'ютерна інженерія»

за освітньо-професійною програмою 123 Комп'ютерна інженерія
(офіційна назва)
на тему «Комп'ютерна система управління персоналом підприємства «ModernDesign» з детальним опрацюванням користувацького інтерфейсу та побудови і налаштування корпоративної мережі»
затверджену наказом ректора НТУ «Дніпровська політехніка» від ______________ №____

Розділ	Зміст	Термін виконання
Стан питання та постановка завдання	На основі матеріалів виробничих практик, інших науково-технічних джерел сформулювати завдання, конкретизувати предмет та мету роботи	18.05.2020
Технічні вимоги до комп'ютерної системи	На основі матеріалів виробничих практик, інших науково-технічних джерел сформулювати технічні вимоги до розробки комп'ютерної системи	25.05.2020
Спеціальна частина	Розв'язати завдання з розробки комп'ютерної системи з опрацюванням побудови та налаштування клієнтської частини системи	01.06.2020
Економічна частина	Економічно обґрунтувати доцільність витрат на створення та дослідження системи	08.06.2020
Охорона праці	Розробити організаційно-технічні заходи, щодо реалізації правил безпеки при експлуатації системи	15.06.2020

Завдання видано ________________ проф. Гнатушенко В.В
(підпис керівника) (прізвище, ініціали)
Дата видачі ________________
Дата подання до екзаменаційної комісії ____________________
Прийнято до виконання ________________ Морозов Б.Д.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 73 с., 38 рис., 11 табл., 1 додаток, 13 джерел.

Об’єкт розробки: комп’ютерна система для управління персоналом для дизайн студії с детальним опрацюванням та налаштуванням користувацького інтерфейсу.

Мета: створення комп’ютерної системи управління персоналом.

Розробка інтерфейсу комп’ютерної системи для управління персоналом орієнтована на поліпшення роботи менеджменту підприємства шляхом спрощення та автоматизації обліку різних етапі робочого процесу підприємства.

Система виконана відкритою, дозволяє здійснювати технічну і програмну модернізацію системи, а так само забезпечує виконання наступних функцій:

- облік даних робітників
- подання заявок на відпустку для робітників
- затвердження відпусток для робітників з боку менеджменту
- контроль робочих процесів персонально для кожного робітника
- додавання нових користувачів в систему управління
- відображення останньої інформації та новин які можуть стати у нагоді робітнику

Розробка комп’ютерної мережі виконана відповідно до завдання на дипломну роботу бакалавра.

Розроблена структура системи і реалізована у вигляді веб-застосунку, перевірено роботу згідно вимог підприємства.

СИСТЕМА, УПРАВЛІННЯ, ПЕРСОНАЛ, ІНТЕРФЕЙС, ВЕБ

ЗМІСТ

ПЕРЕЛІК УМНОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

UI (User's Interface) – користувацький інтерфейс

API (англ. application programming interface) – набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення.

UX (User experience) – загальні враження користувача про зручність використання системи або продукту

AJAX (Asynchronous Javascript and XML) – асинхронный JavaScript та XML, концепція створення користувацьких інтерфейсів для веб-застосунків.

Скрін (від англ. Screen - екран) – найчастіше використовується в значенні одиниці веб-інтерфейсу – сторінки яку бачить користувач.

SPA (Single page application) – веб-застосунок що використовує AJAX для усунення необхідності запитувати у сервера інформацію про веб-сторінку при зміні в інтерфейсі.

MVC (Model View Controller) – шаблон проектування програмної системи.

AWS (Amazon Web Services) – найпоширеніша в світі платформа для хмарних обчислень.

SVG (Scalable Vector Graphics) – масштабована векторна графіка.

W3C (World Wide Web Consortium) – Консорціум Всесвітнього павутиння — головна міжнародна організація, що розробляє й впроваджує технологічні стандарти для всесвітнього павутиння.

DOM (Document Object Model) – об'єктна модель документа, це специфікація для роботи із структурованим документом.

Токен – це електронний ключ для доступу до чого-небудь. В контексті веб-застосунків як правило використовується як засіб автентифікація клієнта сервером.

Хо́стинг (англ. hosting) — послуга, що включає надання дискового простору, підключення до мережі та інших ресурсів для розміщення фізичної інформації на сервері, що постійно перебуває в мереж.

Uptime — це час який комп’ютерна система працювала безперервно та безперешкодно. Вимірюється від запуску системи і до моменту запланованого припинення роботи системи.

JSX (JavaScript XML) — це розширений шаблонний синтаксис за структурою схожий на XML.

EC2 Інстанс — це віртуальний сервер у Amazon Elastic Compute Cloud (EC2) для запуску програм у інфраструктурі AWS.

ВСТУП

З поширенням мережі інтернет все частіше підприємства та звичайні користувачі визнають застарілість інтерфейсів програмних комплексів які розроблені для використання в межах операційної системи як системний програмний застосунок та визнають необхідність уніфікації інтерфейсів у вигляді веб-застосунків.

Розробка інтерфейсів для веб-застосунків це галузь яка на сьогодні є невід'ємною частиною розробки сучасного програмного застосунку. Істотною перевагою побудови веб-застосунків є те, що він буде виправно та стабільно виконувати свої функції незалежно від операційної системи клієнта.

Як і раніше основними інструментами побудови веб-інтерфейсу є HTML, CSS та JavaScript. Але з плином часу та випуском нових стандартів ці інструменти отримали неймовірний розвиток, та продовжують покращуватися. З випуском нових стандартів починають з'являтися і принципово нові підходи та фреймворки для розробки.

В рамках цієї роботи основною мовою розробки буде використано JavaScript оскільки він фактично не має аналогів. В якості основної бібліотеки буде використано React.

1 СТАН ПИТАННЯ І ПОСТАНОВКА ЗАВДАННЯ

1.1 Обґрунтування необхідності системи

Підприємство що замовило систему це невелика компанія з головним офісом у Германії, що займається професійною розробкою дизайнерських продуктів та брендуванням для клієнтів по всьому світу на замовлення. Вони не мають власних серверних та будь-яких мережевих пристроїв на балансі підприємства, бо це економічно не вигідно, тому використовують хмарні технології, переважно від компанії Google та Amazon. Платформа AWS від Amazon буде використано для зберігання ресурсів пов’язаних с розроблюваною системою.

Увесь штат компанії працює віддалено, тому в компанії немає централізованих робочих місць, проте кожен робітник має вільний та швидкий доступ до мережі інтернет.

Керуюча ланка компанії складається с 10 менеджерів, які безпосередньо керують 75 працівниками, переважно дизайнерами. В планах компанії закладено зріст не менше ніж на 30% щорічно в найближчі 3 роки. Територіально робітники можуть бути розташовані в будь-якій точці світу де є доступ до мережі інтернет.

Для будь-якого підприємства його працівники є основним ресурсом. Завдання менеджменту – проводити ефективне управління ресурсами компанії. Важко переоцінити необхідність для менеджменту наявності зручних інструментів для управління персоналом. В даному випадку бізнесу, в зв’язку з постійним розширенням та віддаленою формою роботи, вкрай необхідно отримати гнучкий інструмент для управління персоналом у вигляді веб-застосунку.

Наразі вони не мають чіткого, єдиного методу відстеження відпусток, задач та стану своїх робітників. Існуючи методи у вигляді групових чатів у мессенджерах та табличок – незручні та ненадійні, а системи на ринку розроблені для великих корпорацій наповнені непотрібним для малого бізнесу функціоналом, з ціновою політикою яку малий бізнес не може собі дозволити.

Також самим робітникам часом важко налагодити ефективну взаємодію адже, так як всі працюють віддалено більшість робітників не мають контактних даних та не знають до кого з колег можна звернутися, та з яким питанням.

Система що буде реалізована в цій роботі покликана вирішити наведені проблеми для цього бізнесу, та водночас вона достатньо узагальнена та може буде використана на будь-якому підприємстві зі схожими потребами та бізнес процесами. Реалізація веб інтерфейсу в свою чергу є невід’ємною частиною розробки такого веб-застосунку та основним напрямком цієї роботи.

1.2 Робочі процеси підприємства

На підприємстві замовника системи більшість робітників виконують завдання від керівництва які отримують у вигляді повідомлень, або підчас відео дзвінків. Відстежити виконання таких задач, часом дуже складно. Також складно оцінити продуктивність робітника та скласти звіт не маючи конкретної інформації про те який об’єм роботи за який час було виконано. Цю проблему вирішить система автоматизованого трекінгу задач для робітників.

Відпустки на підприємстві відслідковуються з використанням звичайних таблиць та видаються тільки шляхом прямого діалогу між керівництвом та робітником, на що необхідно виділити час який можна було б використати більш продуктивно. Автоматизована система відпусток вирішить це питання. Робітники зможуть створювати заяви на відпустки в онлайн форматі, та своєчасно отримувати інформацію щодо статусу своєї заявки.

Так як компанія не має єдиного офісу для всіх працівників люди переважно контактують лише з тими, з ким співпрацюють тісно по роботі. Тому якщо виникає ситуація коли необхідно зв’язатись з кимось в компанії, такі питання доводиться вирішувати через керівництво, такий процес є докорінно неправильним адже покладає на керівництво багато додаткової роботи яку можна автоматизувати за допомого єдиної та доступної бази для всіх робітників, де можна швидко знайти необхідні контакти та інформацію про колегу.

1.3 Аналіз існуючих систем управління персоналом

На рикну існує декілька схожих за напрямком продуктів. Більшість яких не мають гнучких налаштувань і пропонуються як єдина система для великих корпорацій, з повним пакетом можливих додатків та розширень. Через це вони мають відповідно високу ціну, яка не підходить для малого бізнесу. Далі розглянемо два найбільш схожих продукти які частково виконують ті функції, що й розроблювана система.

Система «Hurma» має наступний функціонал:

- Робота з кандидатами та вакансіями
- Планування інтерв’ю та тестових завдань
- Керування тестовим періодом та його оцінка
- Лікарняні, відпустки, відгули
- Повідомлення про події у компанії
- Управління цілями і ключовими результатами

– Планування та статистика

Можна відмітити що в цій системі база робітників доступна лише керівництву та рекрутерам і весь основний функціонал націлений на набір нового персоналу та завдання іміджу компанії. Цінова політика на продукт динамічна, залежить від кількості співробітників у компанії яка буде використовувати продукт та відображена у таблиці 1.1. Приклад інтерфейсу системи зображено на рисунку 1.1. Система розроблена для використання на Українському ринку. Має лише три варіанти локалізації – Українська, Російська та Англійська.

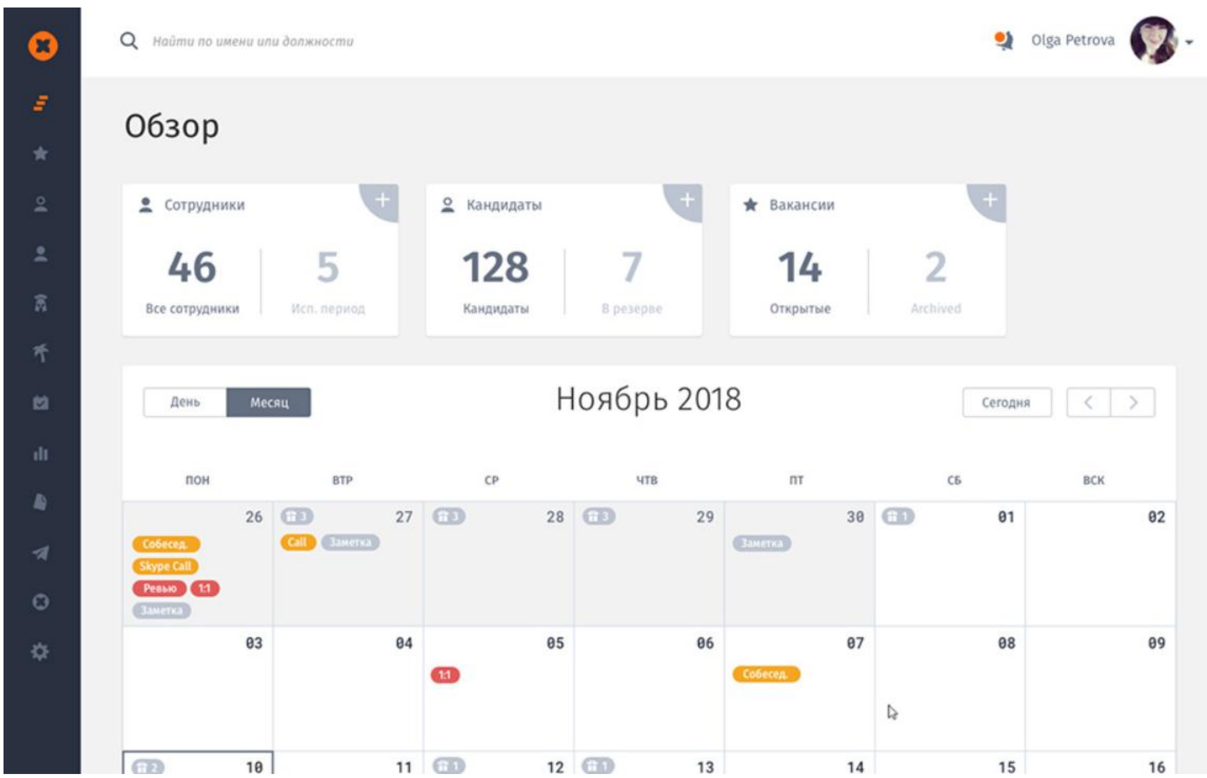

Рисунок 1.1 – Приклад інтерфейсу системи «Hurma»

Таблиця 1.1 – Цінова політика на продукт «Hurma»

Кількість співробітників	Ціна грн./міс
До 30	2300
До 50	4050
До 100	6500
До 150	8700
До 200	9500
До 250	12150
До 300	14550
До 950	40000
Більше 1000	Індивідуальні умови

Система «Atlassian JIRA». Розроблена компанією Atlassian в 2002 році. Доступна в двох версіях: хмарній і серверній. Зараз JIRA включає в себе три проекти: JIRA Software (для розробників), JIRA Service Desk (підтримка проекту), JIRA Core (управління проектами), кожен з яких можна купити окремо. JIRA спрямована на керування великими проектами ентерпрайз рівня у сфері розробки програмного забезпечення хоча може буди використана як система управління персоналом.

Доступний функціонал:

- Планування задач
- Відстеження задач

– Налаштування полів у задачах під будь-які типи процесу розробки
– Складання звітів
– Інтеграція з системами контролю версій
– Мобільній застосунок

В системі відсутній контроль відпусток та єдина відкрита база працівників. Ціна починається від 190 гривень за кожного нового користувача. Система не має жодних специфічних для будь-якого регіону функцій, всі функції доступні незалежно від країни та регіону використання системи. Також доступний широкий вибір локалізацій. Приклад інтерфейсу системи зображено на рисунку 1.2.

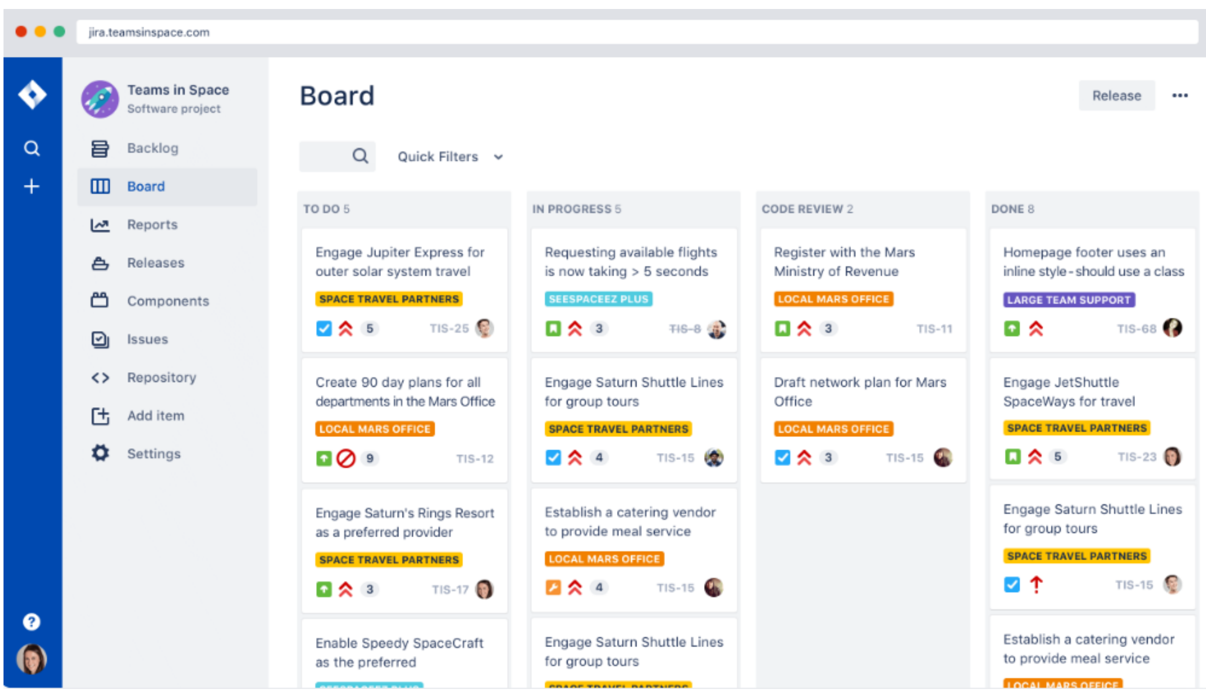

Рисунок 1.2 – Приклад інтерфейсу системи «JIRA»

В цілому можна сказати що основні недоліки готових систем такі:

– Багатий функціонал вимагає навчання, підготовки та налаштування перед використанням
– Цінова політика розрахована лише на великі підприємства

– Відсутня система обліку відпусток з доступом для всіх робітників

1.4 Стислі відомості про сучасні веб технології

Вже багато років HTML невід'ємною частиною розробки веб-застосунків. За багато років він пережив багато оновлень і зараз представлений у виді свого останнього стандарту HTML5. Цей стандарт був розроблений великою групою до якої увійшли AOL, Apple, Google, IBM, Microsoft, Mozilla, Nokia, Opera та кілька сотень інших виробників. Специфікації HTML5 не обмежуються тільки розміткою і включають в себе низку веб-технологій, котрі у сукупності формують відкриту веб-платформу — програмне оточення для роботи кросплатформових застосунків, здатних взаємодіяти з обладнанням, і які підтримують засоби для роботи з відео, графікою і анімацією, що надає розширені мережеві можливості. Найважливішими оновленням в останній специфікації є наступні:

- Підтримка тегів для відео та аудіо матеріалів
- Збереження даних на стороні клієнта Session Storage, Local Storage, IndexedDB
- Векторна графіка з використанням SVG та canvas

Офіційно останньою доступною специфікацією CSS на сьогодні є CSS3. CSS це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду. Самі ж сторінки написані мовами розмітки даних. CSS є основною технологією всесвітньої павутини, поряд із HTML та JavaScript. Як і у випадку с HTML специфікації для CSS створюються W3C. Найважливішими оновленням в останній специфікації є наступні:

- Функція calc()

- Медіа-запити для екрану та сенсору
- Псевдоселектори :valid, :invalid та :empty
- Селектор за сусіднім елементом
- Алгебраїчні вирази в nth-child()

Для JavaScript останньою версією стандарту наразі є ECMAScript 9 затверджений міжнародною організацією ECMA згідно зі специфікацією ECMA-262. Однак найбільш поширеною у використанні на сьогодні є версія ECMAScript 6. Нижче у таблиці 1.2 буде наведено основні зміни у версіях починаючи з ECMAScript 6 до ECMAScript 9.

Таблиця 1.2 – Основні відмінності у стандартах ECMAScript.

Назва	Дата випуску	Зміни від попереднього видання
ECMAScript 6	Червень 2015 року	Додано let і const. Додано значення параметрів за замовчуванням. Додано Array.find (). Додано Array.findIndex ().
ECMAScript 7	Червень 2016 року	Зміни до блоку-області видимості змінних і функцій, оператор зведення в ступені для чисел. await, async ключові слова для асинхронного програмування.
ECMAScript 8	Червень 2017 року	Додано додавання рядків. Додані нові властивості об'єкта. Додані функції Async. Додана спільна пам'ять.
ECMAScript 9	Червень 2018 року	Додані властивості rest/spread. Додано асинхронну ітерацію. Додано Promise.finally(). Доповнення до RegExp.

Для створення SPA буде використано бібліотеку React. React (старі назви: React.js, ReactJS) — відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників

React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC, таких як AngularJS. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови веб-застосунків. Як бібліотеку інтерфейсу користувача React найчастіше використовують разом з іншими бібліотеками, такими як Redux.

Для комунікації с сервером буде використано протокол HTTP другої версії – HTTP/2. HTTP/2 був розроблений 2014 року і стандартизований в 2016 році, він базується на попередній версії HTTP 1.1, яка була стандартизована RFC 2616 в 1999 році.

Робочою групою були сформульовані наступні цілі розробки протоколу HTTP/2:

- Створення механізму узгодження який дозволить клієнтам і серверам обирати між HTTP 1.1, 2.0, або потенційно інший не-HTTP протокол.
- Підтримка високорівневої сумісності з HTTP 1.1, наприклад, за допомогою методів, кодів стану, URI, і більшості заголовків.

- Зменшення затримок з метою прискорення завантаження веб-сторінок браузерами. За допомогою стиснення даних у заголовках HTTP, конвеєризації запитів. Також усунення такої проблеми HTTP 1.x, як блокування черги запитів найпершим запитом, та Мультиплексування кількох запитів у одному TCP-з'єднанні
- Підтримка загальноприйнятих існуючих способів використання HTTP, таких як звичайні і мобільні веб-браузери, веб-API, веб-сервери різного масштабу, проксі-сервер, зворотні проксі-сервери, файрволи, мережі доправлення і розповсюдження інформаційного наповнення.

1.5 Коротка характеристика комп’ютерної системи, що розробляється

Система буде розроблена у вигляді веб-застосунку за технологією SPA, буде доступна користувачам будь-яких сучасних пристроїв з виходом у інтернет – настільні ПК, ноутбуки, планшети та смартфони.

Мова інтерфейсу, на вимогу замовника, та для розширення потенційного ринку використання системи, обрана найпоширеніша, міжнародна – Англійська.

Функціонал системи управління персоналом для малого підприємства не залежить від конкретного спрямування бізнесу хоч і розробляється спеціально для дизайнерської студії. Функціонал може бути умовно поділений на чотири основних блоки за напрямками:

1. Автентифікація та авторизація
2. Перегляд даних інших користувачів, редагування власник даних
3. Система обліку відпусток
4. Трекінг задач

Перший блок – це автентифікація та авторизація. В системі можливість додавати нових користувачів матиме лише керівництво, тобто в застосунку буде

систему закритої реєстрації. Автентифікація це процес перевірки даних, таких як токен або ім'я користувача або ідентифікатор користувача та пароль, щоб підтвердити особу. Потім система перевіряє, чи належать облікові дані саме вам.

Авторизація відбувається після успішної автентифікації вашої особи, та необхідна для отримання доступу до усіх функцій застосунку. Під час авторизації визначається до яких ресурсів має доступ автентифікований користувач.

Другий блок – це надання можливості усім користувачам мати доступ до даних своїх колег або підлеглих. Ці дані можуть включати:

- Ім’я та прізвище співробітника
- Посада
- Контактний номер телефону
- Контактна адреса електронної пошти
- Дата народження
- Фотокартка співробітника
- Дата наступної відпустки
- Для керівників також можливе відображення встановленої поточної заробітної платні та індикацію психологічного стану співробітника

Ці дані необхідні в системі для налагодження ефективного робочого процесу, комунікації і обліку даних про співробітника. Виконання системи у вигляді своєрідно внутрішньої соціальної мережі для працівників також може виконувати роль іміджевої складової для нових співробітників та сприятиме формалізації відносин між колегами.

Третій блок – це відпустки для робітників. Кожен робітник на своїй персональній сторінці зможе зробити запит на відпустку. Цей запит буде відправлено до менеджера який зі свого боку вирішує чи може він підтвердити заплановану відпустку згідно з робочим планом підприємства або потрібно зв’язатися с працівником для внесення корекції. Менеджеру доступний список усіх запланованих відпусток для працівників у цей час кожний працівник може відавши персональну сторінку колеги дізнатися чи є в нього заплановані відпустки.

Четвертий блок – постановка та відстеження задач у роботі. Менеджер може створювати задачі та встановлювати відповідального за їх виконання. Створена задача зберігає наступні дані:

- Назва задачі
- Стислий опис задачі
- Посилання на персональну сторінку користувача що створив задачу
- Виконавця задачі, та посилання на його персональну сторінку
- Дата створення
- Поточний статус виконання задачі (задача готова до роботи, задача в роботі, виконання задачі заблоковано або задача завершена)

Таким чином завжди можна перевірити над якою задачею конкретно працює співробітник, побачити її статус, змінити її статус. Швидко скласти звіт на основі відображених в системі даних.

2 ТЕХНІЧНІ ВИМОГИ ДО СИСТЕМИ

2.1 Вимоги до системи в цілому

З боку замовника до системи було висунуто ряд вимог:

– Система має бути виконана у вигляді веб-застосунку, та інтегрована в хмарну архітектуру AWS
– Безперешкодний доступ до системи має надаватися через мережу інтернет у будь-який час (uptime ~99%)
– Накладні розходи на утримання системи не повинні перевищувати 500 доларів США на рік, у перший рік використання. Обслуговування системи повинно бути автоматизоване та включено у суму накладних витрат
– Необхідно використовувати єдиний дизайн код у всіх компонентах застосунку
– Мова інтерфейсу системи – Англійська.
– Система має бути протестована розробником за основними користувацькими сценаріями
– Система має бути розширювана як у плані збільшення користувачів та навантаження у декілька разів, так і мати можливість безперешкодного доопрацювання окремих елементів у майбутньому

– Розробка за системою graceful degradation, тобто підтримка широкого профілю пристроїв та поступове спрощення функціоналу для продовження підтримки
– Функціонал повинен включати:
    1. Автентифікацію, авторизацію та створення нових користувачів
    2. Перегляд та редагування даних для авторизованих користувачів
    3. Систему обліку відпусток
    4. Трекінг задач

2.2 Сценарії взаємодії з системою

2.2.1 Реєстрація та авторизація

На рисунку 2.1 показано сценарії взаємодії користувача з системою на етапі реєстрації та авторизації. Користувач може бути направлений на сторінку зміни паролю або пройти автентифікацію.

Рисунок 2.1 – Сценарії взаємодій для користувача

На рисунку 2.2 показано сценарії взаємодії менеджера з системою на етапі реєстрації та авторизації. Менеджер має можливість створити нових користувачів і так само як і звичайний користувач може бути направлений на сторінку зміни паролю або пройти автентифікацію.

Рисунок 2.2 – Сценарії взаємодій для менеджера

2.2.2 Перегляд даних

На рисунку 2.3 показано сценарії взаємодії користувача з системою на етапі перегляду даних. На цьому етапі користувач має можливість переглянути список задач, їх статуси, назву, опис, дізнатися хто створив задачу та хто відповідальний за виконання. Також необхідно мати можливість переглянути свої відпустки, відпустки колег так їх дати та статуси. Користувач може переглядати власні облікові дані, так змінювати свій номер телефону та рік народження. Так само користувач використовуючи пошук може дізнатися необхідні дані про колег, переглянути фотокартку чи контактні дані.

Рисунок 2.3 – Сценарії взаємодії користувача за системою на етапі перегляду даних

На рисунку 2.4 показано сценарії взаємодії менеджера з системою на етапі перегляду даних. Менеджер має ті самі можливості в системі на етапі перегляду даних що і звичайний користувач, єдина різниця в даному випадку це мотиви для перегляду.

Рисунок 2.4 – Сценарії взаємодії менеджера з системою на етапі перегляду даних

2.2.3 Облік відпусток

На рисунку 2.5 показано сценарії взаємодії користувача з системою на етапі обліку відпусток. Користувач може створити запит на відпустку с бажаними датами, відслідковувати статус погодження відпустки у керівництва, та отримувати останні дані про відпустки колег щоб мати змогу планувати ефективну взаємодію базуючись на датах відпусток.

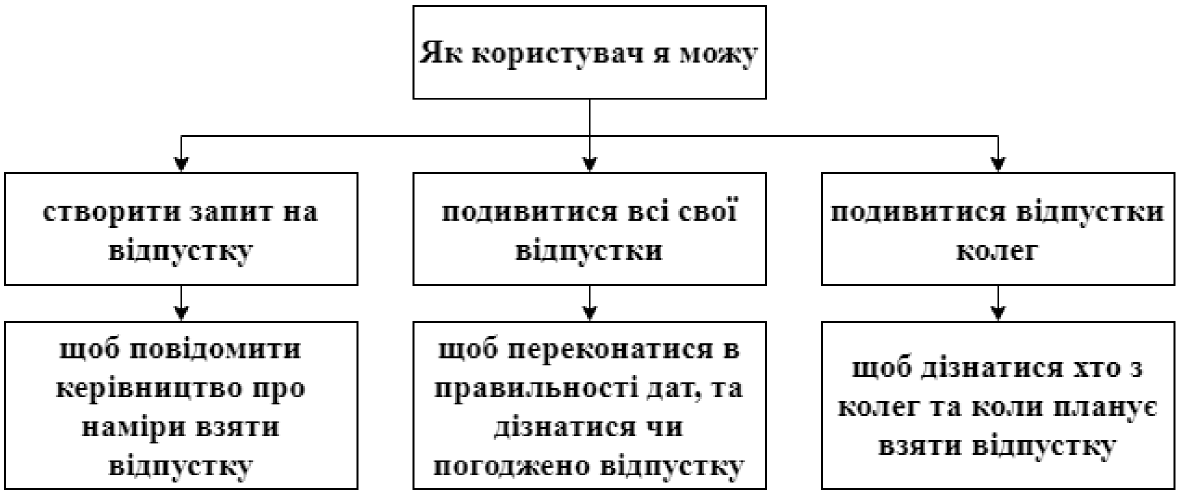

Рисунок 2.5 – Сценарії взаємодії користувача з системою на етапі обліку відпусток

На рисунку 2.6 показано сценарії менеджера користувача з системою на етапі обліку відпусток. Менеджер може створити запит на відпустку с бажаними датами, відслідковувати статус погодження власної відпустки, та отримувати останні дані про відпустки колег щоб мати змогу затвердити або відмінити запитану відпустку. Також за необхідності запит на відпустку користувача може бути видалено.

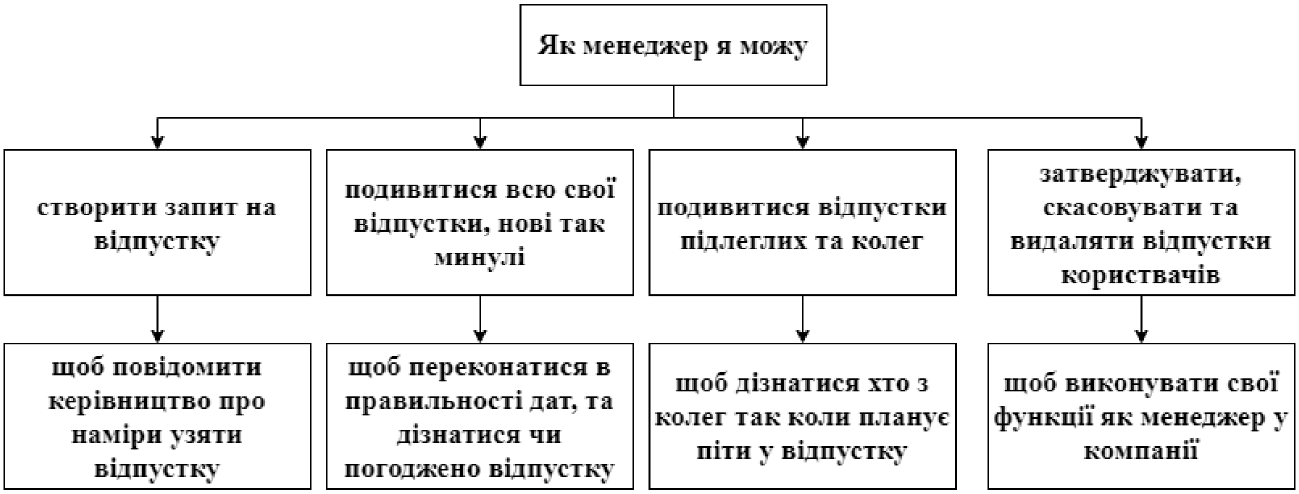

Рисунок 2.6 – Сценарії взаємодії менеджера з системою на етапі обліку відпусток

2.2.4 Трекінг задач

В розділі трекінгу задач можливості менеджера та звичайного користувача однакові, тому на рисунку 2.7 показано сценарії взаємодії користувача або менеджера з системою на етапі роботи із задачами. Кожен користувач має змогу створити нову задачу та відкоригувати вже існуючу задачу створену іншим користувачем. Змінити статус виконання та відповідального за виконання задачі.

Рисунок 2.7 – Сценарії взаємодії з системою на етапі роботи із задачами

2.3 Проектування графічного інтерфейсу

Графічний інтерфейс додатку має три основних підрозділи: авторизація, профіль користувача, сторінка задачі та сторінка відпусток. В свою чергу вони мають бути поділені на менші частини,таким чином маємо наступну структуру інтерфейсу:

Авторизація:

- Сторінка входу – «Login»
- Сторінка створення користувача – «Add user»
- Сторінка відновлення паролю – «Password reset»

Профіль користувача:

- Блок з даними про користувача – «Profile Card»
- Блок індикації настрою – «How Do you Feel»
- Блок нещодавніх змін – «Recent Changes»
- Блок відпусток – «Vacations Card»
- Блок допомоги – «Help Card»
- Блок задач – «Tasks»
- Блок слайдів з новинами компанії – «News Slider»

Сторінка задачі:

- Діалог створення задачі – «Task create modal»
- Заголовок – «Title»
- Власник– «Owner»
- Опис – «Description»
- Поле завантаження файлів – «Attachments»
- Час останньої зміни – «Latest update»

– Відповідальний за задачу – «Assignee»

– Статус задачі – «Status»

Сторінка відпусток:

– Діалог створення відпустки – «Vacation create modal»

– Фільтр – «Show all/pending/my»

– Список відпусток – «Vacations List»

Кожний зазначений компонент поділено на ще менші частини, такі як поля вводу, кнопки, блоки тексту, іконки тощо.

3 РОЗРОБКА ЗАСТОСУНКУ

3.1 Вибір технологій та їх обґрунтування

3.1.1 Вибір платформи для застосунку

Аналізуючи вимоги поставлені під час проектування застосунку можемо виявити що однією з пріоритетних задач є доступність додатку на максимальній кількості пристроїв. Для виконання такої задачі найбільш підходить веб-додаток. Оскільки використовувати веб додатки можна практично на будь-якому пристрої з доступом для мережі інтернет, інакше кажучи вони вкрай невибагливі до середовища запуску.

Платформою для запуски мобільного застосунку є браузери, основні сучасні браузери це:

– Google Chrome

– Mozilla Firefox

– Safari
– Microsoft Edge (Chromium-based)

Google Chrome – це крос платформний веб-браузер розроблений корпорацією Google. Перший запуск версії для Windows відбувся у вересні 2008 року, пізніше розробку було перенесено на Linux, IOS, macOS та Android. Та відносно нещодавно відбувся запуск Chrome OS – операційною системи де Google Chrome є одним із ключових компонентів усієї системи. Використовує механізм WebKit для обробки веб сторінок. На даний час доступно 47 мов інтерфейсу, та за приблизною оцінко він займає 68% ринку браузерів за використанням.

Mozilla Firefox – це крос платформний веб-браузер, з відкритим вихідним кодом розроблений Mozilla Corporation. Перший запуск версії для Windows відбувся у вересні 2002 року. Використовує механізм Gecko для обробки веб сторінок. Доступний на 90 мовах світу, та займає приблизно 9% світового ринку браузерів за використанням.

Safari – веб-браузер розроблений компанією Apple, браузер за замовчування для всіх пристроїв компанії. Перший запуск відбувся 7 січня 2003 року на платформу OS X (зараз Mac OS). Пізніше відбувся запуск версії для Windows, але у 2012 році розробники відмовилися від подальшої підтримки на інших платформах, тому зараз актуальні версії доступні для систем IOS та Mac OS. Safari використовує механізм WebKit для обробки веб сторінок. На даний час доступно 100 мов інтерфейсу, та за приблизною оцінко він займає 9% ринку браузерів за використанням.

Microsoft Edge – веб-браузер розроблений компанією Microsoft як браузер за замовчуванням для операційної системи Windows. Використовує механізм Blink для обробки веб сторінок. До недавнього часу не мав популярності через відсутності багатьох можливостей для розробки або некоректно їх реалізовував, проте нещодавно Microsoft зробили важливий крок та перенесли Microsoft Edge на відкриту платформу Chromium, що дозволить радикально спростити його підтримку та покращити функціонування. Доступний на 96 мовах світу, та, не дивлячись на популярність Windows як операційної системи, за приблизними оцінками займає лише до 6% світового ринку браузерів за використанням.

3.1.2 Веб-застосунки та нативні застосунки

Розглянемо переваги використання веб-застосунків на відміну від нативних застосунків.

Нативний застосунок – це застосунок розроблений з використанням інструментів які розроблено спеціально для розробки під конкретну платформу або систему. Розробляючи нативний застосунок розробник завжди має максимально безпосередній доступ до налаштування усіх деталей, програмних та апаратних функцій пристрою. Наприклад використовуючи мову Swift для написання додатку для операційної системи IOS розробник може отримувати доступ до зчитування показників датчиків пристрою, використання камери або динаміків.

Основні плюси нативних застосунків:

– Максимальна ефективність

У разі розробки нативного додатку під кожну систему окремо розробник завжди має можливість вирішити поставлено задачу так я було задумано розробником апаратної частини пристрою, це як правило, завжди максимально ефективний шлях, до того ж це зменшує вірогідність збоїв у роботі програми в майбутньому.

– Уніфікація UX

В розробника завжди є доступ до вже доборе знайомих користувачам компонентів інтерфейсу платформи, які використовуються за замовчуванням в цільовій системі. Це спрощує сприйняття додатку для користувача та знижує необхідність розробки додаткових допоміжних матеріалів для ознайомлення користувача із системою.

– Повна підтримка додатка розробником апаратного пристрою

Всі системні покращення та оновлення з боку розробника системи де встановлено нативний застосунок будуть автоматично впливати, та покращувати роботу застосунка. Наприклад випуск нової версії операційної системи для телефону в якому пришвидшено роботу ядер відповідальних за обробку зображення, автоматично покращить та пришвидшіть роботу відповідних частин застосунку.

Однак у разі необхідності підтримки застосунку на двох і більше операційних системах або навіть платформах, наприклад комп’ютерів на операційний системі Linux та Windows та смартфонів на IOS та Android. В даному випадку для розробки нативного додатку для кожної системи треба було б залучити щонайменше чотири спеціаліста до кожної платформи та системи, відповідно використати більше коштів та витратити щонайменше удвічі більше часу на розробку усієї системи під кожну платформу та операційну систему.

Веб-застосунок – це програмний застосунок що виконує більшу частину операцій на сервері, а результат основних обчислень та операцій відображає користувачу через інтерфейс використовуючи веб-браузер. Такому застосунку як правило здебільшого неважлива середа виконання, адже більшість операцій специфічних для системи виконуються або на сервері або з використанням узгоджених API, що мають спільний програмний інтерфейс на усіх платформах де представлені веб-браузери.

Веб-застосунки мають наступні переваги:

– Охоплення максимально можливого спектру користувачів

За рахунок того що веб-застосунок невибагливий до середовища виконання він може бути запущений на будь-якому пристрої де є веб-браузер, що в сьогоднішній реальності майже будь-який побутовий комп’ютер, планшет або смартфон

– Швидкість розробки

При розробці багато деталей та особливостей різних платформ можна не брати до уваги. Слід лише розуміти відповідні відмінності веб-браузерів та можливі різновиди фінальної імплементації деяких механізмів в цих браузерах. Немає потреби диференціювати кодову базу під різні пристрої, один і той самий програмний код буде запущено для всіх пристроїв. Це спрощує внесення будь яких змін або виправлень у програмний код.

– Легше розгортання застосунку

Для того щоб фінальний користувач отримав доступ наприклад до мобільного застосунку, його розробник повинен сплатити фіксовану плати за доступ до магазину на цільовій платформі, та під час випуску першої та кожної наступної версії, потрібно проходити перевірку коду, що може займати час та ускладнює доступ для додатку. В свою чергу для того щоб отримати доступ до веб-застосунку необхідно лише мати веб-браузер, його як правило вже встановлено на кожному пристрої за замовчуванням.

3.1.3 Вибір фреймворку

Необхідність використання фреймворку для розробки обґрунтовано тим що наразі не є доцільним відмовлятися від вже готових та оптимізованих імплементацій базових речей у застосунку. Наприклад розробка з нуля стратегії відстеження змін для застосування оновлень у DOM дереві, алгоритму між сторінкового роутингу з побудовою історії або налаштування потоку даних в застосунку – витрата часу на ретельну проробку базових механізмів може призвести до невиправданих фінансових так часових витрат.

Наразі існує два популярні фреймворки які можуть виконати поставлені задачі – це React та Angular. Розглянемо переваги та недоліки обох фреймворків.

React – це фреймворк розроблений компанією Facebook. На початку використовувався в середині компанії для розробки таких продуктів як «Instagram» та «WhatsApp». Основною відмінністю є те ще він контролює лише частину відображення (View) у моделі Model View Controller, завдяки чому є гнучким інструментом для якого модель та контролер можуть бути реалізовані окремо власноруч або з використанням іншим бібліотек. Дає можливість самостійно обирати структуру та ієрархію компонентів застосунку. Має наступні переваги:

– Віртуальне DOM дерево

Якщо вам потрібно змінити адресу користувача - яка записана десь між блоком коду HTML - віртуальний DOM розглядає лише різницю коду між попередніми та поточними тегами HTML. Потім він змінює лише ту частину, де потрібно застосувати оновлення. Ця функція підвищує ефективність React під час роботи з великим набором даних.

– Односторонній потік даних

У цьому типі прив'язки даних користувач змінює стан моделі, який надає зміну елемента інтерфейсу користувача, але якщо змінити елемент інтерфейсу, стан моделі не зміниться сам по собі. На рисунку 3.1 показано спрощену схему потоку даних у React.

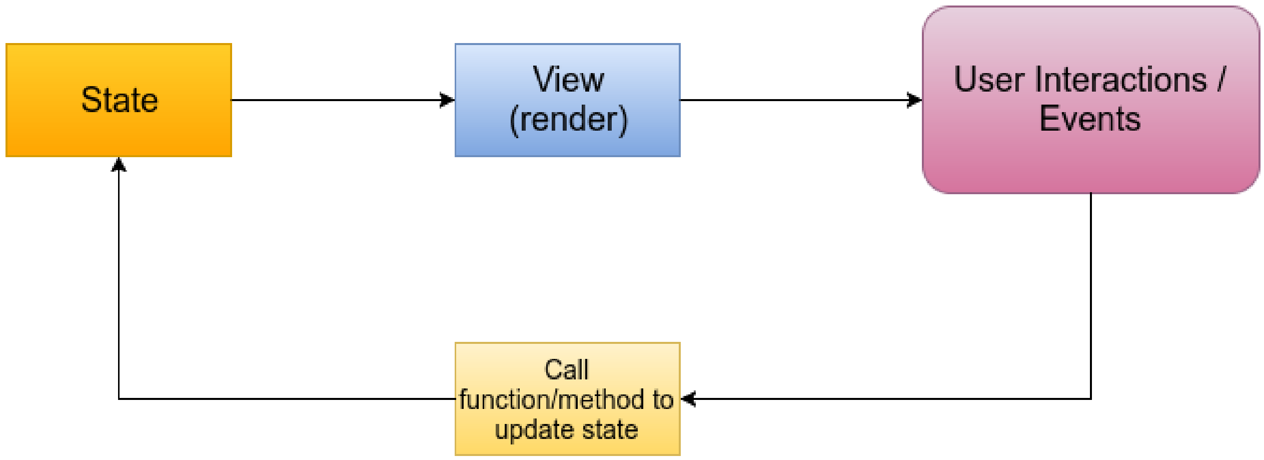

Рисунок 3.1 – Схема потоку даних у React

– JSX замість атрибутів та шаблонів

Замість використання шаблонів React використовує JSX, що є розширенням для XML, та вбудовує його у JavaScript.

– Кросплатформні мобільні застосунки

Для розробки гібридних мобільних додатків React використовує платформу під назвою React Native - яку також розробляє Facebook. React Native дозволяє користувачеві створювати власні компоненти так само як і React для веб. Такий застосунок може бути встановлений безпосередньо у смартфон.

Angular – це фреймворк, що підтримується та розробляється Google. Він використовується в декількох додатках Google, включаючи Google Analytics і Firebase Console. Відмінністю є те ще він повноцінно покриває всі ланки MVC підходу. Буквально диктує розробнику те, як програма має бути структурована. Angular також забезпечує набагато більше функцій без використання додаткових бібліотек включаючи ін’єкцію залежностей, шаблони, маршрутизацію тощо.

Серед інших відмінностей це:

– Реальне DOM дерево

Використання звичайного DOM дерева призводить до оновлення всієї структури дерева тегів HTML, перед тим як оновити цільовий тег. Ця особливість негативно впливає на ефективність Angular при роботі з великим набором даних.

– Двосторонній потік даних

Двостороння прив'язка даних означає, що якщо користувач вносить зміни в елемент UI, то відповідна модель також змінюється. Якщо користувач змінює безпосередньо модель, інтерфейс користувача також автоматично вносить відповідні зміни. На рисунку 3.2 показана схема взаємодії у застосунку за MVC.

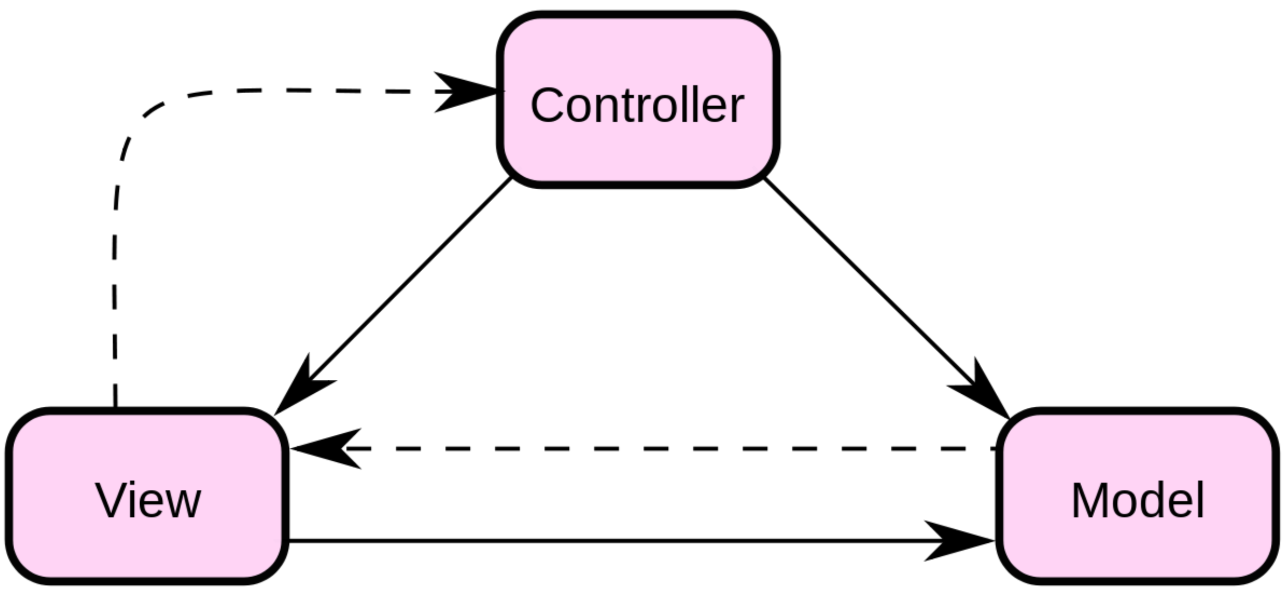

Рисунок 3.2 – Схема взаємодії компонентів у застосунку за MVC

– Кросплатформні мобільні застосунки

Для розробки гібридних мобільних додатків Angular використовує ionic, він забезпечує потужну бібліотеку компонентів інтерфейсу. На виході отримаємо звичайний веб застосунок запущений у WebView.

Тож оцінивши відмінності двох фреймоврків найкращим архітектурним рішенням було обрано React. Такий вибір базується в основному на тому що React використовує віртуальне DOM дерево, що є додатковим шаром оптимізації та має одно направлений потік даних що спростить розробку. На рисунку 3.3 показано етапи оновлення дерев компонентів при використанні віртуального DOM дерева.

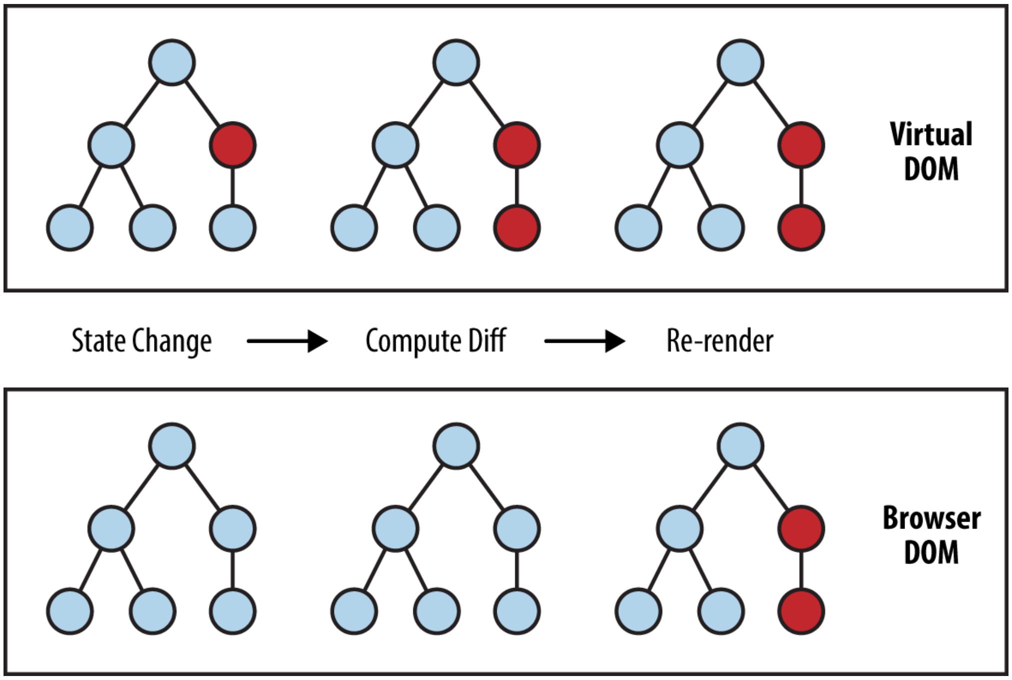

Рисунок 3.3 – Етапи оновлення дерев компонентів

В іншому фреймворки дуже схожі за своєю реалізацією, тому в даному випадку вибір можна було б зробити базуючись лише на персональний прихильності бо відмінності в фінальному результаті незначні для розроблюваної системи.

3.2 Розробка апаратної частини комп’ютерної системи

Так як для хостингу системи використовуються хмарні рішення від AWS в апаратній частині буде розглянуто вибір правильного типу хмарного сховища. В AWS сервіс що дозволяє використовувати хмарні комп’ютери для обчислень має назву «Amazon Elastic Compute Cloud» або скорочено «EC2». В нашому випадку для front-end частини застосунку необхідно обрати EC2 інстанс на якому можна буде запустити програмне забезпечення HTTP сервера, що буде по запиту видавати нам необхідні частини додатку. У якості програмного забезпечення HTTP серверу буде використано Nginx, операційна система інстансу Ubuntu Server 16.04 LTS. За своїм призначенням інстанси EC2 умовно поділяються на:

– Інстанси загального призначення
– Інстанси оптимізовані для обчислень
– Інстанси оптимізовані для пам’яті

Розроблювана система не має складних обчислень і не оперує великими об’ємами даних, тому для цього проекту найкращим вибором буде інстанс загального призначення. AWS пропонує цілий ряд різних типів інастансів загального призначення, далі стисло розглянемо деякі їх особливості.

A1 – дозволяють істотно скоротити витрати і ідеально підходять для горизонтально масштабованих робочих навантажень. Базується на архітектурі Arm. Інстанси A1 - це перші інстанси EC2 на базі процесорів AWS Graviton з 64-бітними ядрами Arm Neoverse і спеціальними кремнієвими мікросхемами, розробленими AWS. Доступна кількість ядер центрального процесору від 1 до 16, кількість ОЗУ від 2 до 32 гігабайт, пропускна здатність мережі до 10 гігабіт на секунду. Прикладами використання можуть бути: масштабування робочих навантажень, наприклад веб-серверів, мікросервісов на базі контейнерів, платформ кешування і розподілених сховищ даних, а також середовищ розробки.

T3 – це нове покоління універсальних інстансів зі змінюваною продуктивністю. При використанні надають продуктивність ЦПУ базового рівня з можливістю підвищити її в будь-який момент на будь-який необхідний період часу. Коли робоче навантаження знаходиться нижче базового порогового значення, інстанси T3 накопичують кредити ЦПУ. Один кредит ЦПУ надає інстансу T3 можливість при необхідності на одну хвилину підвищити продуктивність ядра ЦПУ до максимальної. У безлімітному режимі інстанси T3 можуть в будь-який момент підвищити продуктивність на будь-який необхідний період часу. Сімейство процесорів що використовуються це Intel Scalable (2,5 ГГц). Доступна кількість ядер центрального процесору від 2 до 8, кількість ОЗУ від 0,5 до 32 гігабайт, пропускна здатність мережі до 5 гігабіт на секунду. Прикладами використання можуть бути: мікросервіси, інтерактивні додатки з низькою затримкою, невеликі і середні бази даних, віртуальні робочі столи, середовища розробки, репозиторії коду і критично важливі для бізнесу додатки.

T2 – це інстанси зі змінюваною продуктивністю, які забезпечують базовий рівень продуктивності ЦПУ з можливістю його підвищення. Безлімітні інстанси T2 можуть підтримувати високу продуктивність ЦПУ до тих пір, поки цього вимагає робоче навантаження. Для більшості робочих навантажень загального призначення безлімітні інстанси T2 забезпечують достатню продуктивність без додаткової плати. Якщо потрібно виділити більше ресурсів ЦПУ на інстанси протягом тривалого часу, це також можна зробити з оплатою за фіксованим тарифом: 5 центів за годину роботи віртуального ЦПУ. Сімейство процесорів що використовуються це Intel Xeon. Доступна кількість ядер центрального процесору від 1 до 8, кількість ОЗУ від 0,5 до 32 гігабайт, пропускна здатність мережі від 1 до 5 гігабіт на секунду. Прикладами використання можуть бути: сайти та інтернет-додатки, середовища розробки, сервери збірки, репозиторії коду, мікросервіси, середовища тестування, а також додатки для бізнесу.

Окрім A1, T2, та T3 AWS має ще цілий ряд рішень на основі високопродуктивних процесорів AMD серії EPYC 7000, Intel Xeon E5‑2686 v4 та Intel Xeon Platinum де під віртуальні машини виділяється до 96 ядер процесору та до 384 гігабайта ОЗУ та виділена мережа з пропускною здатністю до 25 гігабіт на секунду. Проте такі рішення значно в багато разів перевищують необхідні нам параметри для застосунку.

Якби ми мали необхідність обрати лише один інстанс для всіх частин застосунку (клієнт частина, серверна частина, база даних та сховище файлів) ми мали б обрати інстанс типу T2. Проте виходячи з кращих архітектурних практик ми розподілимо усі компоненти системи. HTTP сервер, що обробляє запити до інтерфейсу буду запущено окремо на інстансі типу A1.

3.3 Опис реалізації основних компонентів застосунку

3.3.1 Побудова та запуск застосунку

Для менеджменту залежностей в застосунку буде використано пакетний менеджер NPM (Node package manager). В конфігураційному файлі визначеному наступні необхідні точки входу до побудови застосунку:

- «yarn start» для запуску локального серверу для потреб розробки
- «yarn build» для побудови оптимізованої збірки для використання на сервері
- «yarn install» для встановлення або оновлення пакетів

Для розробки буде використано ряд бібліотек, їх також додано до списку залежностей у файлі «package.json», список основних використаних бібліотек для фінального застосунку наведено в таблиці 3.1. В таблиці 3.2 наведено список бібліотек що використовуються лише на етапі розробки.

Таблиця 3.1 – Список основних використаних бібліотек

Назва пакету	Версія
axios	0.19.2
classnames	2.2.6
debounce	1.2.0

moment	2.25.3
react-router-dom	5.1.2
react	16.13.1
semantic-ui-react	0.88.2

Таблиця 3.2 – Список бібліотек що використовуються на етапі розробки

Назва пакету	Версія
eslint-config-airbnb	18.1.0
eslint-config-prettier	6.10.1
eslint-plugin-jsx-a11y	6.2.3
eslint-plugin-prettier	3.1.3

У файлі «yarn.lock» зберігаються згенеровані кодом списки залежностей пакетів які встановлено.

Складено файл «.gitignore» для виключення непотрібних файлів із індексу системи контролю версій git. Непотрібні файли це як правило файли що стосуються налаштування середи розробки або файли де можуть зберігатися секретні ключі.

Для збереження читабельності та стилістичної корректності коду відповідно міжнародних стандартів було створено файл «.eslintrc» в якому задано

Змінні оточення що використовуються при побудові збірки застосунку повинні бути вказані у файлі «.env», приклад конфігурації файлу викладено у файлі «.env.default». Додано лише дві змінних:

– «REACT_APP_API_URL» для налаштування адреси базової адреси API сервера
– «REACT_APP_VERSION» для того щоб відрізняти між собою версії додатку які вже на сервері. Цю змінну буде додано до верхньої частини інтерфейсу

Також було складено стислу проектну документацію у файлі «README.md», там стисло викладено основну інформацію про застосунок, доступні команды, та інструкцію для завантаження додатку на сервіс AWS.

На рисунку 3.4 показано структуру директорій проекту. В кореневій директорії розташовані конфігураційні файли проекту, директорія «src» розміщує вже безпосередньо вихідний код застосунку, поділений на семантичні блоки з відповідними найменуваннями.

Рисунок 3.4 – Структура файлів проекту

Збірка застосунку виконується за допомогою пакету webpack. Після запуску команди «yarn build» застосунок збирається у статичні файли html, css, js та файли ресурсів. Для запуску достатньо використати будь який HTTP сервер, що на запит поверне клієнту файл «index.html», подальші файли будуть завантажені вже безпосередньо кодом застосунку. Завдяки мінімізації та оптимізації остаточна версія збірки застосунку займає лише 7.5 мегабайт дискового простору, включаючи усі текстові файли та зображення.

3.3.1 Реєстрація та авторизація

Реєстрація та авторизація в застосунку виконується з використанням сервісу AWS Cognito.

AWS Cognito – це хмарний сервіс менеджменту користувачів. Його задача у тому щоб створити пул юзерів, для проходження авторизації для подальшого доступу до закритої частини застосунку. В якості унікального ідентифікатора користувача виступатиме e-mail.

Першим етапом необхідно щоб адміністратор створив нового користувача. Для цього було розроблено сторінку, що має відносну адресу «/adduser». На рисунку 3.5 показано зовнішній вигляд форми створення користувача.

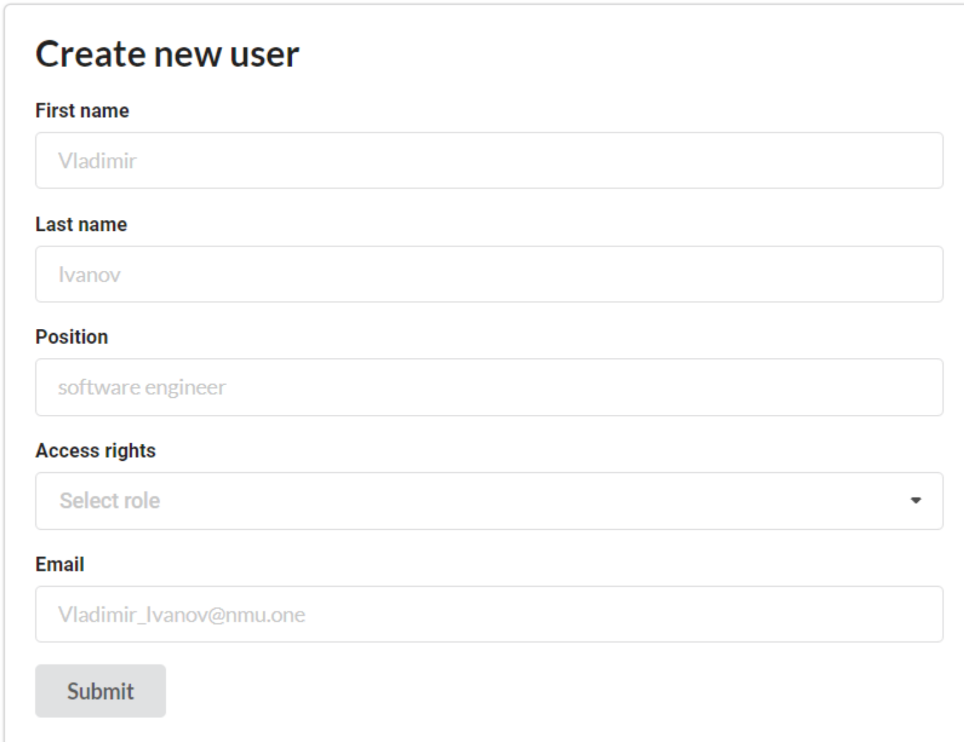

Рисунок 3.5 – Форма створення користувача

Поля «First name», «Last name», «Position», «Access rights», «Email», є обов’язковими для заповнення. Після заповнення полів, на адресу «/signup» на сервері буде відправлено запит. Приклад такого запиту зображено на рисунку 3.6.

Рисунок 3.6 – Формат запиту на створення користувача

Після успішної обробки запиту користувач отримає на казаний email свій тимчасовий пароль. Який разом з email може бути використаний на сторінці, входу, що має відносну адресу «/login». На рисунку 3.7 показано зовнішній вигляд форми входу користувача.

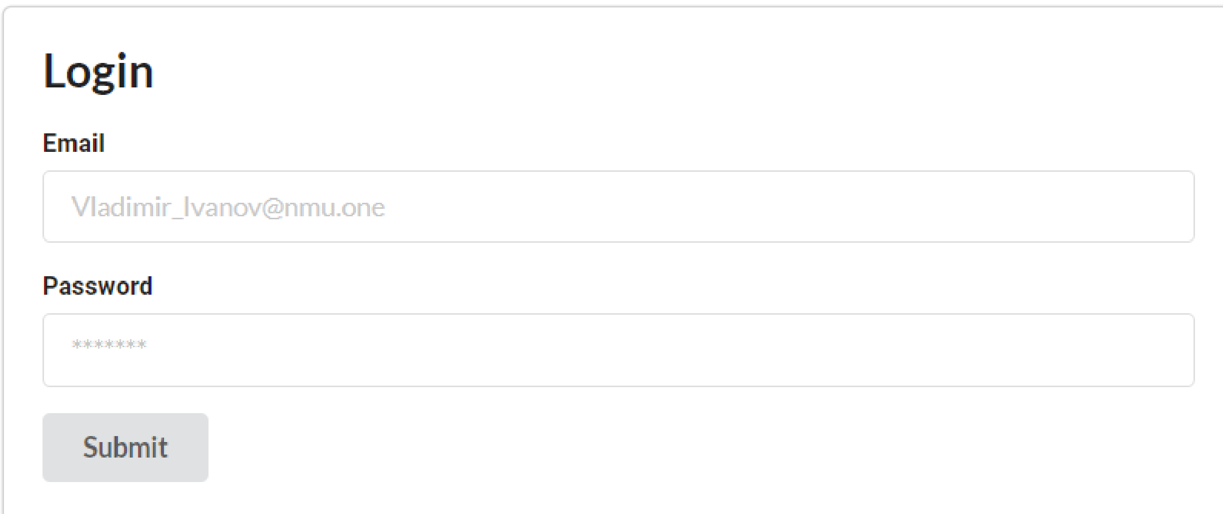

Рисунок 3.7 – Форма входу користувача

Після заповнення полів, на адресу «/signin» на сервері буде відправлено запит. Приклад такого запиту для логіну користувача зображено на рисунку 3.8.

Рисунок 3.8 – Формат запиту на автентифікацію користувача

У відповідь на запит сервіс AWS Cognito надсилає клієнту токен який зберігається в cookies. У подальшому усі інші запити будуть перевіряти чи є цей токен та чи він актуальний. Перший раз, коли користувач скористається тимчасовим паролем, сервер недійшле у відповідь запит на зміну паролю, тому користувача буде направлено на сторінку зміни паролю. Сторінка зміни паролю має відносну адресу «/reset». На рисунку 3.9 показано зовнішній вигляд форми зміни паролю користувача.

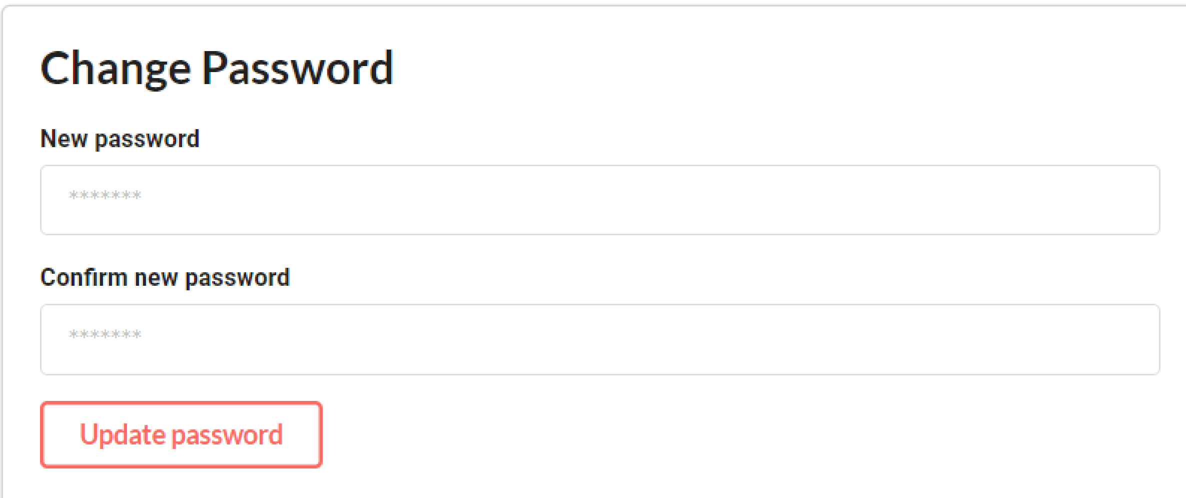

Рисунок 3.9 – Форма зміни паролю користувача

Після заповнення полів, на адресу «/password/required» на сервері буде відправлено запит. Приклад такого запиту для логіну користувача зображено на рисунку 3.10.

Рисунок 3.10 – Формат запиту на зміну паролю користувача

Використовуючи кнопку «Sign Out» в шапці сайту користувач може відправити запит виходу на AWS Cognito. Такий запит інвалюдує виданий користувачу токен, всі подальші запити з використанням інвалідованого токену будуть вихилені сервером.

3.3.2 Сторінка користувача

Після успішної авторизації користувач направляться на свою персональну сторінку. Сторінка розташована за кореневою адресою сервера для власного профілю, або за відносною адресою «/user/id» для того щоб подивитися сторінку користувача за його унікальним ідентифікатором . На рисунку 3.11 показано зовнішній головної сторінки – сторінки користувача. Тут відображені основні інформаційні блоки, блок роботи із задачами та стрічка новин у вигляді зображень.

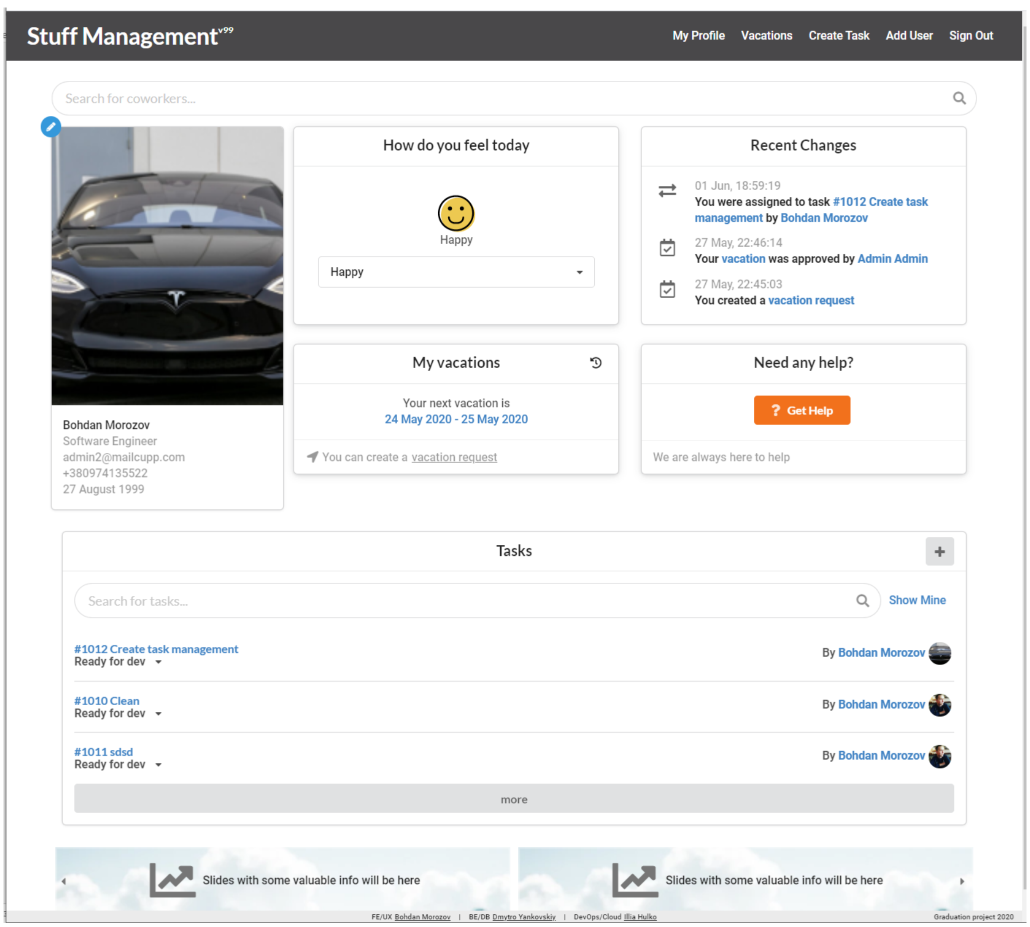

Рисунок 3.11 – Сторінка користувача

Для відображення основної сторінки користувача необхідно зробити п’ять запитів на різні частини контенту.

– «/vacations» для отримання інформації про відпустки користувача
– «/admins» для отримання списку адміністраторів компанії
– «/task» для отримання задач користувача
– «/user» для отримання даних про користувача
– «/recent» для отримання списку нещодавніх змін

Верхню частину головної сторінки займає глобальній пошук серед користувачів. Зовнішній вигляд пошуку показано на рисунку 3.12. При вводі даних з клавіатури через 1000 мілісекунд буду здійснено запит на адресу «/user/search». Який поверне список знайдених користувачів, перші три знайдені користувачі будуть відображені на інтерфейсі як результат пошуку.

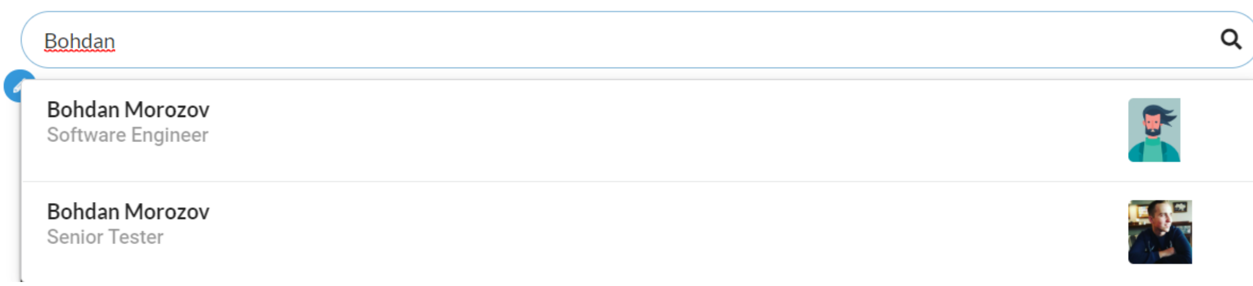

Рисунок 3.12 – Пошук користувачів

Формат запиту на глобальній пошук серед користувачів показано на рисунку 3.13.

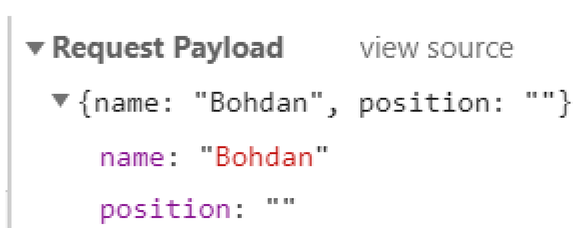

Рисунок 3.13 – Формат запиту на пошук користувачів

На головній сторінці натиснувши на іконку олівця біля блоку свої облікових даних можна потрапити до режиму редагування профілю. Вікно редагування профілю зображено на рисунку 3.14.

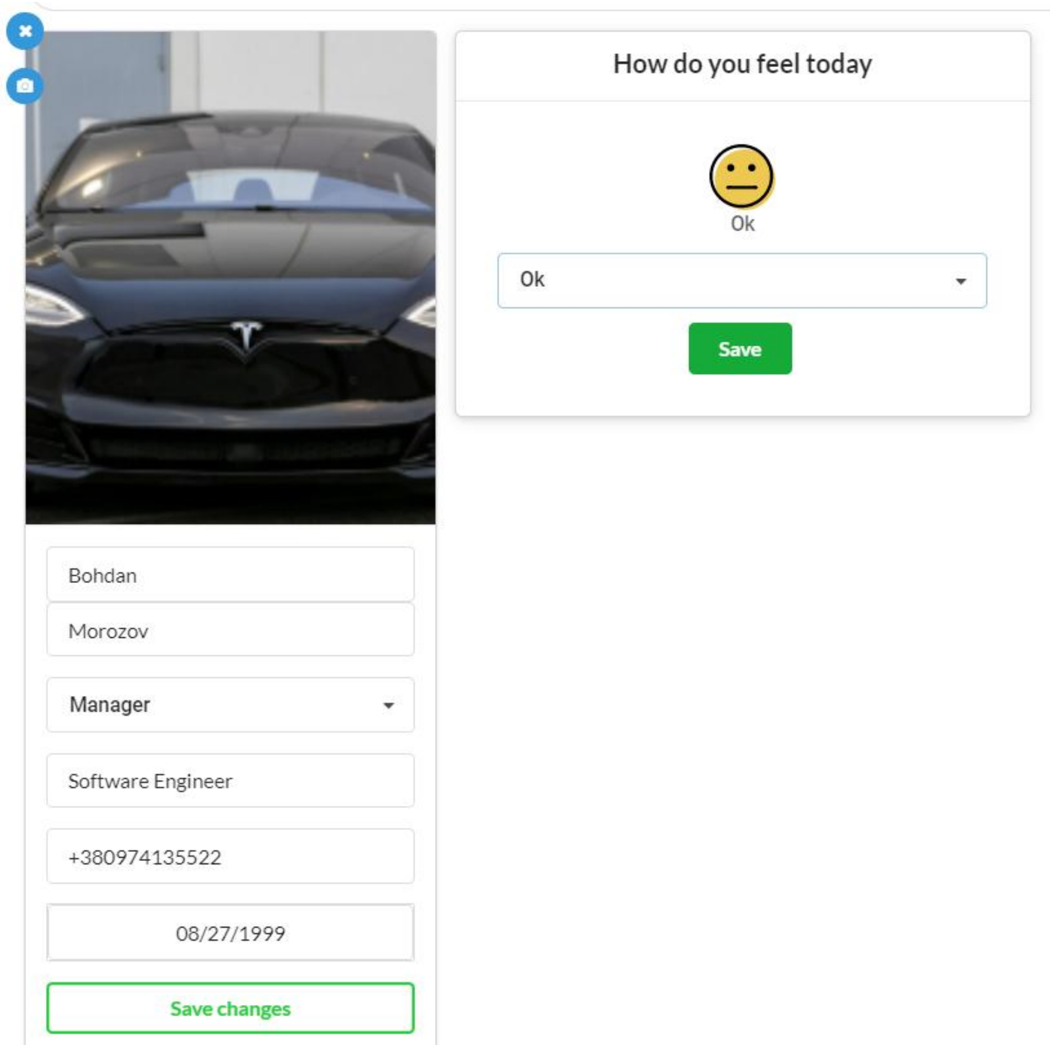

Рисунок 3.14 – Редагування профілю

Після натискання на кнопку «Save Changes» або кнопки «Save», на адресу «/user/id» на сервері буде направлено запит для збереження інформації про користувача. Формат запиту на збереження даних про користувача зображено на рисунку 3.15.

```
▼{position: "Software Engineer", firstName: "Bohdan", lastName: "Morozov", role: "admin",…}
  dateOfBirth: "1999-08-27T09:00:00.000Z"
  firstName: "Bohdan"
  lastName: "Morozov"
  mobilePhone: "+380974135522"
  position: "Software Engineer"
  role: "admin"
```


Рисунок 3.15 – Формат запиту на збереження даних користувача

Натиснувши на іконку фотокамери можна потрапити у вікно оновлення зображення профілю. Зовнішній вигляд вікна показано на рисунку 3.16.

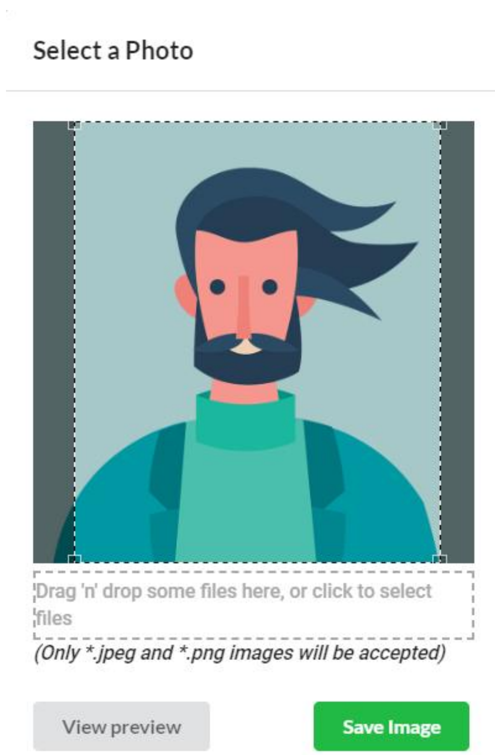

Рисунок 3.16 – Вікно оновлення зображення

Завантаживши зображення, його можна обрізати, але лише у рамках дозволенного співвідношення сторін. При натисканні «Save Image» оновлене зображення буде відправлено на сервер, на адресу «/user/photo» у вигляді двійкового коду. Там воно буде декодовано та збережено у файловому сховищі AWS S3.

У правій частині головної сторінки розташовано блок нещодавніх змін - «Recent Changes». Зовнішній вигляд блоку показано на рисунку 3.17. Він відображає наступні що були зроблені нещодавно:

– користувач був обраний виконавцем задачі
– змінився статус задачі де користувач є виконавцем
– користувач створив запит на відпустку
– запит на відпустку користувача було підтверджено або скасовано

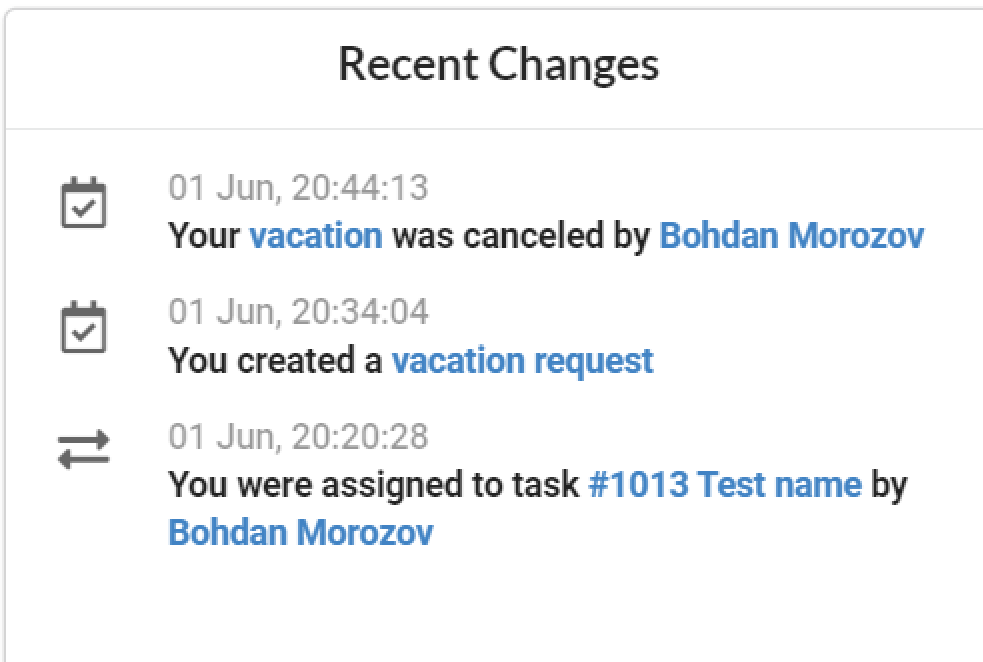

Рисунок 3.17 – Блок нещодавніх змін

Нижче під основними блоками розташовано блок задач. В цьому блоці можна виконувати пошук по задачам, перейти до вікна створення задачі натиснувши іконку зі знаком плюса, оновити статус задачі, перейти на сторінку власника задачі або сторінку самої задачі. Також реалізовано фільтр що відображає лише свої задачі або усі задачі. При натисканні кнопки «more» із фільтром налаштованим на демонстрацію усіх задач буде асинхронно запитано ще три наступні задачі із бази даних. Таку реалізація обрано за огляду на те що може бути створено сотні задач, відображати їх усі одразу величезним списком є недоцільним. На рисунку 3.18 показано вигляд блоку задач на головній сторінці.

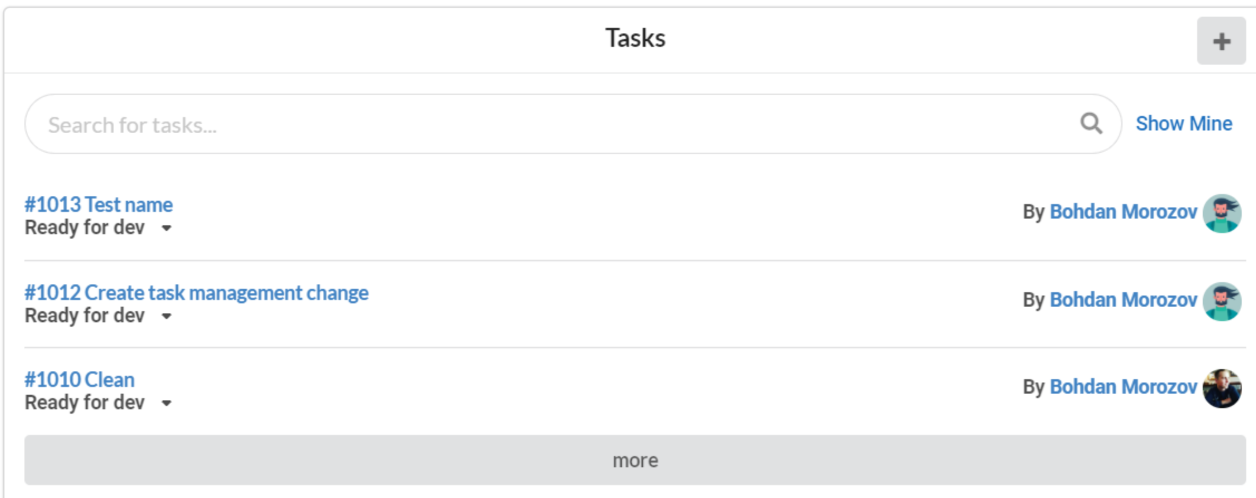

Рисунок 3.18 – Формат запиту на збереження даних користувача

Формат запиту на пошук задач у блоці на головній сторінці показано на рисунку 3.19.

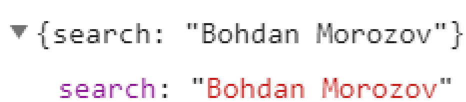

Рисунок 3.19 – Формат запиту на пошук задачі

Формат запиту на довантаження наступного пулу задач до списку показано на рисунку 3.20.

Рисунок 3.20 – Формат запиту трьох наступних задач

3.3.4 Сторінка задачі

Для відображення задачі перш за все необхідно створити її задавши назву, опис та відповідального за виконання. Вікно створення задачі можна викликати із головного меню в верхній частині застосунку або використовуючи блок задач на головній сторінці натиснувши на відповідну іконку. На рисунку 3.21 показано форму створення задачі.

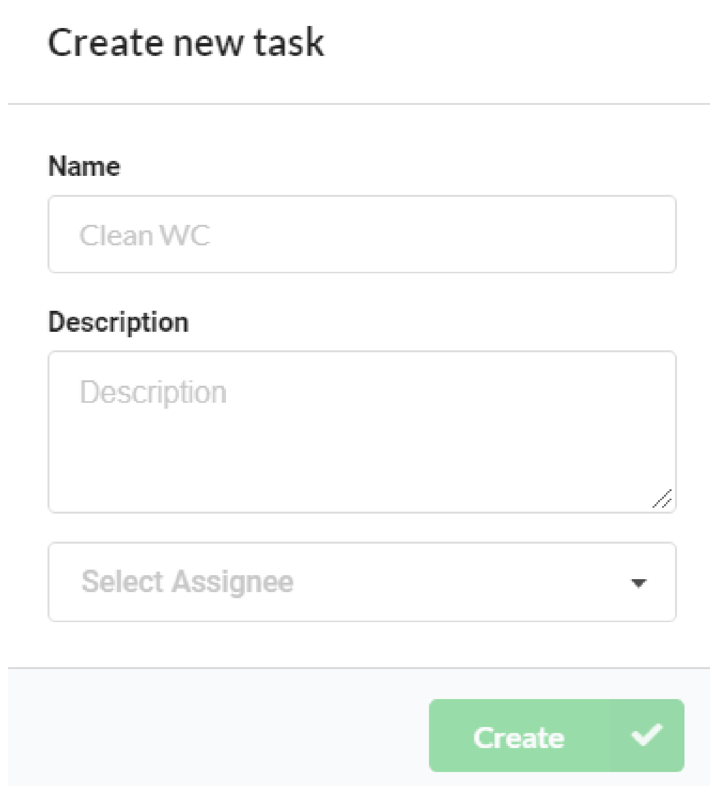

Рисунок 3.21 – Форма створення задачі

Після заповнення полів, на адресу «/task» на сервері буде відправлено запит. Приклад такого запиту для створення задачі зображено на рисунку 3.22.


```
▼{title: "Test name", description: "test description",…}
  assignedID: "e51ac683-7797-4cac-8e27-e6ad934d0a2e"
  description: "test description"
  title: "Test name"
```


Рисунок 3.22 – Формат запиту на створення задачі

Після створення задачі користувач буде відразу направлений на сторінку де вже безпосередньо відображено всю інформацію про щойно створену задачу. Сторінка задачі розташована за відносною адресою «/task/id». На рисунку 3.23 зображено зовнішній вигляд сторінки задачі.

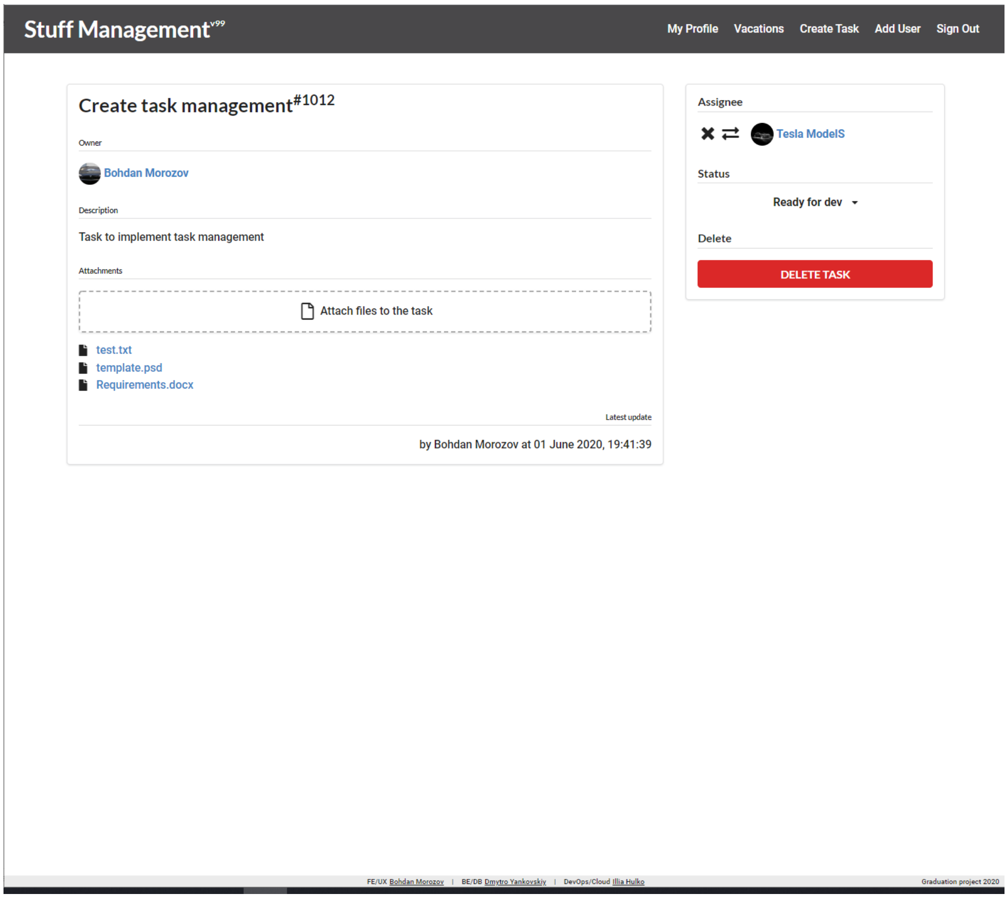

Рисунок 3.23 – Сторінка задачі

Для отримання даних про задачу за її унікальним ідентифікатором використовуємо адресу «/task/id». На сторінці в основному блоці відображено назву задачі, її номер, власника задачі, опис задачі, поле для завантаження файлів, список завантаження файлів, дату останнього оновлення. Завантаження файлів тимчасово відключено для мінімізації витрат на хостинг застосунку. В правому блоці бачимо відповідального за задачу, її статус, та кнопку видалення задачі. Назва, опис, відповідальний користувач та статус задачі можуть бути змінені. Формат запиту на зміну наведено на рисунку 3.24.

```
▼{title: "Create task management change", description: "Task to implement task management↵",…}
  assignedID: "bfd20442-a7f3-4991-9ff1-b0f85eaa0a1c"
  description: "Task to implement task management↵"
  status: "Ready"
  title: "Create task management change"
```


Рисунок 3.24 – Формат запиту на зміну задачі

Після успішного запиту на оновлення, задачу буде одразу оновлено, нові дані будуть відображені моментально, після отримання підтвердження успішної зміни від серверу.

3.3.5 Сторінка відпусток

Для відображення відпусток їх необхідно створити. Вікно створення відпустки можна викликати із головного меню в верхній частині застосунку або використовуючи блок відпусток на головній сторінці. Створені рисунки будуть збережені в базі даних на сервері. На рисунку 3.25 показано форму створення відпустки.

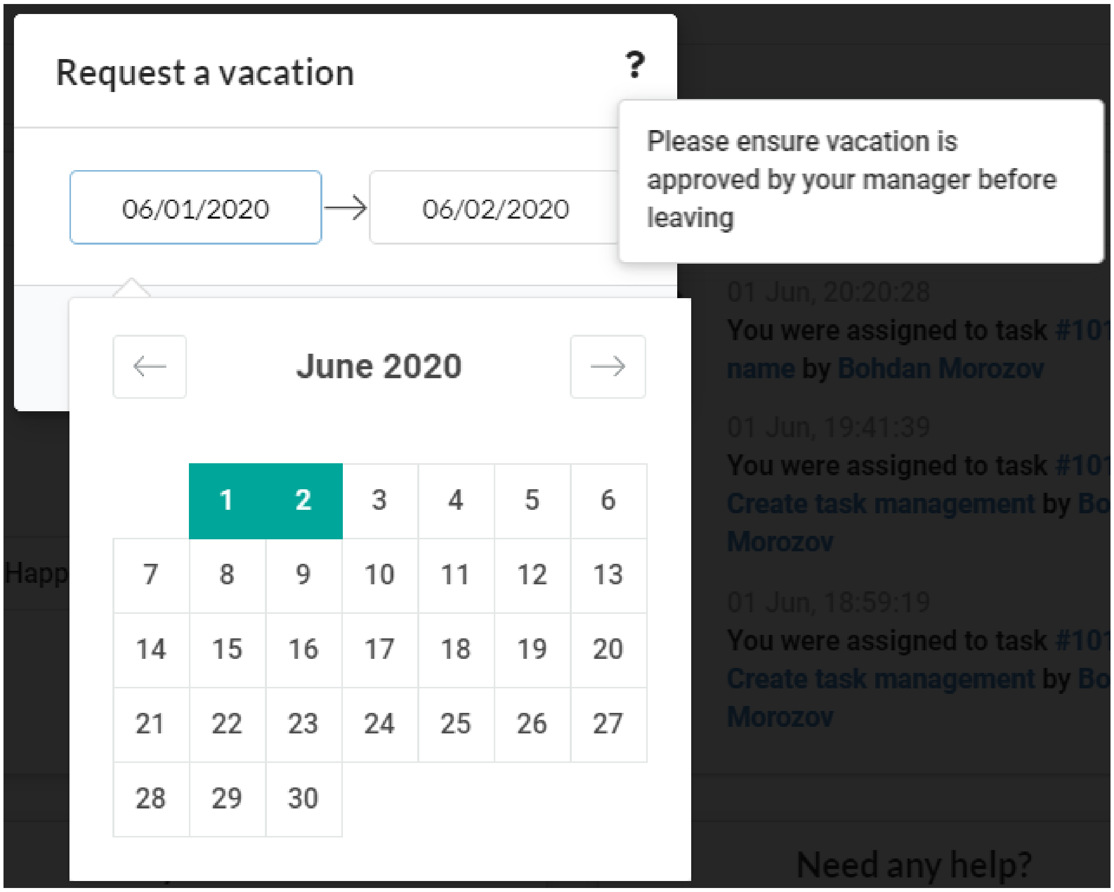

Рисунок 3.25 – Вікно створення відпустки

Для створення необхідно задати бажані дати початку та закінчення відпустки та на натиснути кнопку «Create». На адресу «/vacations» на сервері буде направлено запит. Формат запиту на створення відпустки наведено на рисунку 3.26.


```
▼{startDate: "2020-06-01", endDate: "2020-06-04"}
   endDate: "2020-06-04"
   startDate: "2020-06-01"
```


Рисунок 3.26 – Формат запиту на створення відпустки

Після створення, запит буде доступний на сторінці відпусток. Сторінка відпусток розташована за відносною адресою «/vacations». За адресою

«/vacations/id» можна отримати список відпусток користувача на його унікальнім ідентифікатором. На рисунку 3.27 зображено зовнішній вигляд сторінки відпусток.

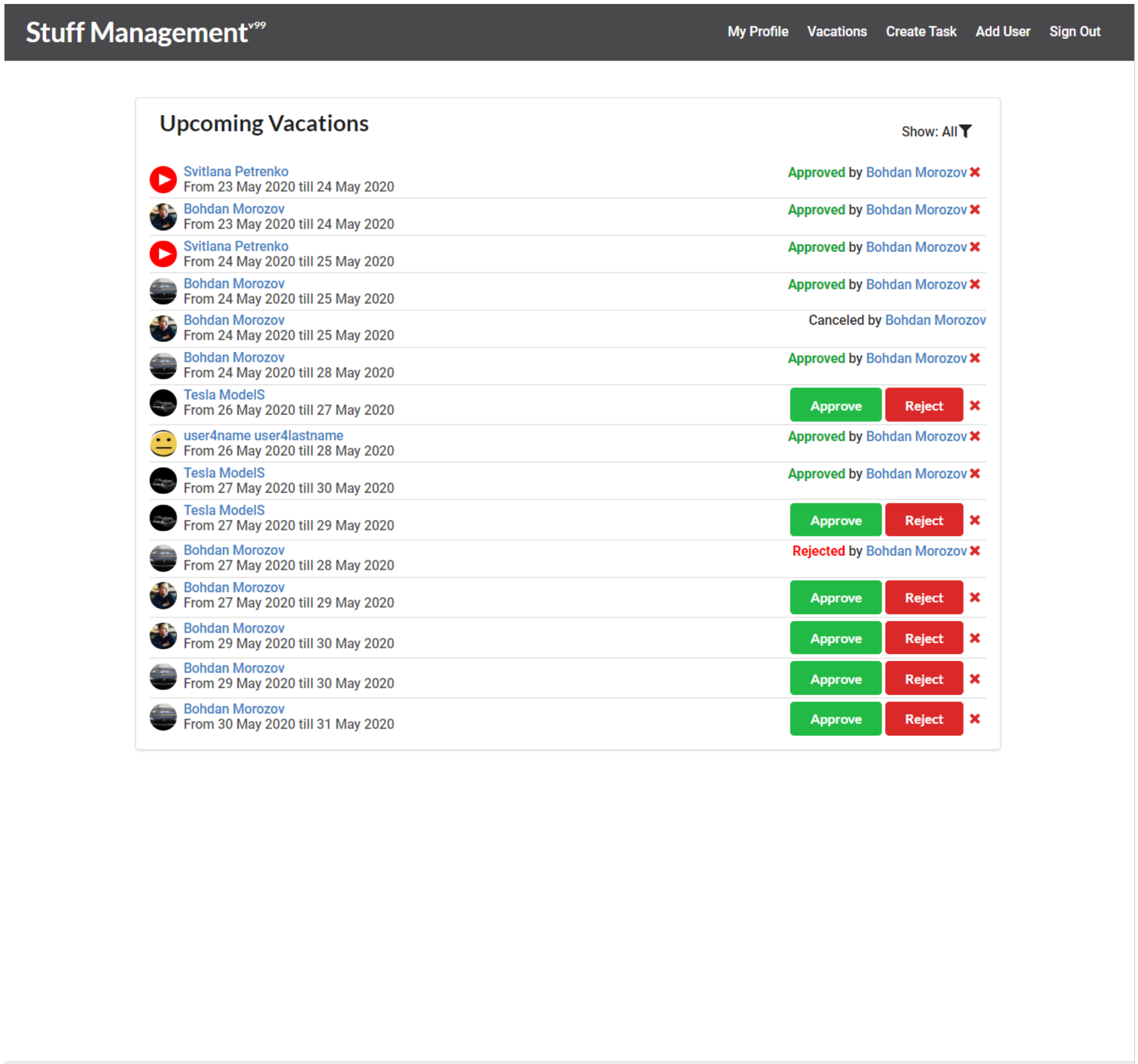

Рисунок 3.27 – Сторінка відпусток

Для отримання даних про відпустки використовуємо адресу на сервері «/vacations». На сторінці за замовчуванням в порядку наближення показні відпустки усіх користувачів. Застосувавши фільтри можна подивитися лише усі відпустки, лише свої відпустки, та відпустки які ще не були відмічені менеджером. Коли менеджер обирає підтвердити або скасувати запит на відпустку, за допомогою кнопок «Approve» або «Reject» на сервер направляється запит. Формат запиту показано на рисунку 3.28.


```
▼{status: "Rejected"}
   status: "Rejected"
```


Рисунок 3.28 – Формат запиту на зміну статусу відпустки

Натиснувши на червоний хрест менеджер може видалити відпустку, так відпустка вважається відміненою.

4 ЕКОНОМІЧНА ЧАСТИНА

Розроблений проект являє собою програмний застосунок вбудований в комп’ютерну систему. Для технічного забезпечення розробки необхідно було використати комп’ютерну техніку та низку канцелярських виробів.

При оцінці фінансових затрат та визначенні результатів було виявлено, основні розділи:

– заробітна плата співробітникам
– витрати на матеріали та комплектуючі
– накладні витрати
– витрати на обслуговування системи

4.1 Зарплата робітникам

Над клієнтською частиною веб застосунку працює лише один співробітник, який виконує роль як розробника так і тестувальника системи. Система розроблювалася за гнучкими методологіями, де бізнес безпосередньо керував напрямом розробки з короткими ітераціями, тому співробітника було найнято у штат, чітких часових обмежень на розробку не було. Фінальний час, витрачений на розробку проекту – 2 місяці, при восьми часовому робочому дні та п’ятиденному робочому тижні.

Витрати на заробітну плату розраховуються за формулою:

$$З = О * М \tag{4.1}$$

де О – місячний посадовий оклад

М – кількість місяців

Місячний посадовий оклад співробітника становить 35000 гривень.

Розрахунок:

$$З = 35\,000 * 2 = 70\,000 \tag{4.2}$$

Також від загальної суми заробітної плати потрібно виділити 14% на додаткові виплати та 22% на соціальне страхування. Кінцевий результат розрахунку витрат на заробітну плату робітникам представлено в таблиці 4.1.

Таблиця 4.1 – Витрати на заробітну плату розробнику

Тип витрат	Сума витрат (грн)
Сума заробітної плати за 2 місяці	70 000
Додаткові виплати та премії (14%)	9 800
	Разом: 79 800
Соціальне страхування (22%)	17 556
	Загальна сума заробітної плати: 97 356

4.2 Витрати на матеріали та комплектуючі

4.2.1 Витрати на обладнання

Для розробки даного програмного забезпечення необхідно придбати або орендувати ноутбук. Так як при оренді на 2 місяці кошти витрачені на обладнання будуть становити, майже, повну ціну комп’ютеру чи ноутбука, з точки зору економічної ефективності, обладнання потрібно купувати, а не брати в оренду.

В ході огляду ринку було обрано обладнання основні параметри яких представлені в таблиці 4.2.

Таблиця 4.2 – Параметри обраного обладнання

№	Модель обладнання	Тип	Основні параметри	Потужність (кВт)	Ціна (грн)
1	LG 27MK600M-W	Монітор	Діагональ дисплея: 27"	0,175	5 399
2	Acer TravelMate P6 TMP648-G2 (NX.VFNEU.002)	Ноутбук	Діагональ дисплея: 14" IPS (1920x1080) Full HD/ / Intel Core i7-7600U (2.8 - 3.9 ГГц) / RAM 16 ГБ / SSD 512 ГБ / nVidia GeForce 940MX, 2 ГБ	0,48	24 999
					Загальна сума: 30 398

4.2.2 Витрати на канцелярські вироби

В таблиці 4.3 представлено список та загальну суму витрат на канцелярські вироби, що знадобилися при проектуванні, розробці та тестуванні системи.

Таблиця 4.3 – Розрахунок витрат на канцелярські вироби

Назва	Кількість	Ціна(грн)	Сума(грн)
Зошит	3	10	30
Маркер	2	10	20
Ручка	2	10	20
			Сума: 70

4.3 Накладні витрати

4.3.1 Послуги інтернет зв'язку

Впродовж усієї розробки використовувся високошвидкісний доступ до інтернету, на швидкості до 1 гігабіту на секунду. Провайдером послуги у місті Дніпро є компанія «Тріолан». Аналогічних послуг за адресою де проводилася розробка не надає жодна інша компанія у місті. Щомісячна плата за безлімітний тариф становить 300 гривен. Виходячи з цього можна розрахувати витрати за інтернет на період розробки:

$$\mathrm{I} = 300 * 2 = 600 \text{ грн} \tag{4.3}$$

4.3.2 Витрати на електроенергію

В продовж всього часу розробки використовувались різні електричні прилади, що зумовлює витрати на електроенергію.

Розрахунок місячних витрат на електроенергію можна провести за допомогою формули:

$$Е = П * Т * Ц \tag{4.4}$$

де П – потужність обладнання в кВт

Т – час роботи електроприладу

Ц – ціна за один кВт*год електроенергії, грн

На сьогодні тарифікація електроенергії в Україні становить:

- за обсяг, спожитий до 100 кВт·час електроенергії на місяць – 90,0 коп.
- за обсяг, спожитий понад 100 кВт·час електроенергії на місяць – 168,0 коп.

Результати споживання електроенергії згідно з обраною конфігурацією об-ладнання та чинною тарифікацією представлено в таблиці 4.5.

Таблиця 4.5 Розрахунок витрат на електроенергію

№	Назва обладнання	Потужність(кВт)	Час роботи	Витрати(грн)
1	Монітор	0,175	320	95
2	Ноутбук	0,48	320	259
Загальна сума : 354				

4.4 Витрати на обслуговування системи

Робота системи не потребує наявності у компанії мережевого обладнання або співробітників. Система працює у хмарному сервісі AWS, тому основні витрати на обслуговування як раз пов’язані із оплатою послуг. Проте на перший рік в ціна обслуговування може бути вищою, за рахунок можливих недоліків у системі знайдених лише у процесі експлуатації. Тому на перший рік необхідно ще закласти близько 10% відсотків на наладку або незаплановані зміни у системі після вводу в експлуатацію, такі додаткові трати поширюються лише на перший рік обслуговування.

Витрати на хмарні послуги можна вирахувати помноживши вартість інстансу на годину, на приблизну кількість годин роботи системи. В нашому випадку система повинна працювати постійно, без відключень, тому можна розрахувати витрати за наступною формулою:

$$С = Ц * 8766 \text{ год.} \tag{4.5}$$

де Ц – ціна роботи інстансу за годину

Для розрахунку звернемося до цінової політики сервісу AWS. Обраний нами инстанс типу A1, у варіанті a1.medium при сплаті на годину коштує 0.69 грн. на годину. Проте за передплатою на три роки існує інша пропозиція за якою остаточна ціна за годину становитиме 0.35 грн. З формули маємо що річна сума витрат на сервіси AWS складає 3068 гривень. У таблиці 4.6 викладено всі річні витрати на обслуговування у перший рік.

Таблиця 4.6 – Витрати на обслуговування у перший рік

Тип витрат	Сума витрат(грн)
Сервіси AWS	3 068
Зміни під час експлуатації(10%)	12 877
	Загальна сума: 15 945

Тож маємо що в перший рік обслуговування системи коштуватиме компанії 15945 гривень, та лише 3068 гривень у наступні роки.

4.5 Сумарні витрати

Сумарні витрати на розробку програмного продукту можна побачити в таблиці 4.7.

Таблиця 4.7 – Сумарні витрати на розробку ПЗ

Тип витрат	Сума витрат(грн)
Заробітна плата розробника	79 800
Соціальне страхування	17 556
Покупка обладнання	30 398
Канцелярські вироби	70
Послуги зв’язку	600
Витрати на електроенергію	354
	Загальна сума: 128 778

Таким чином, за комплексним розрахунком, на розробку клієнтської частини комп’ютерної системи загалом було витрачено 128778 грн, що у рази дешевше ніж викуп уже готових системи або ліцензійна плата за готові рішення.

Висновок

Створення клієнтської системи коштуватиме 128778 гривень, створення серверної частини 73309 гривен. Створення та три роки обслуговування системи коштуватиме компанії 256100 гривень. В свою чергу аналогічна розглянута система «hurma» коштувала б щонайменше 313200 гривень. При цьому, у разі розширення компанії, сума продовжувала би збільшуватися, в той час коли розроблена системи не вимагає додаткових трат у зв’язку з кадровим розширенням.

5 ОХОРОНА ПРАЦІ, ПРОМИСЛОВА БЕЗПЕКА ТА ЦИВІЛЬНИЙ ЗАХИСТ

5.1 Аналіз шкідливих і небезпечних вражаючих факторів

При розробці та роботі з програмним забезпеченням необхідно вживати заходів щодо послаблення негативних ефектів. При роботі за комп’ютером зашкодити робітнику можуть одразу декілька факторів, як і обладнання за яким він працює так і умови праці у цілому, робоче місце та приміщення. У таблиці 5.1 наведені небезпечні так шкідливі фактори що можуть вплинути на робітника під час роботи.

Таблиця 5.1 – Аналіз шкідливих і небезпечних вражаючих факторів

№ п/п	Найменування ШНВФ	Джерела ШНВФ	Нормуючи
1	Можливість ураження електричним струмом	Електропроводка, блок живлення персонального комп’ютера	ДЕСТ 12.1.038-82. Електробезпека. Гранично допустимі значення напруг дотику і струмів
2	Тепловиділення від устаткування	Персональний комп'ютер, периферійні пристрої	СанПиН 2.2.4.548-96
3	Недостатня освітленість робочої зони	Робоче місце	СНіП ІІ-4-79
4	Специфічний характер зорової роботи	Монітор, робоче місце оператора	СанПин 2.2.2/2.4.1340-03

		ПЕОМ	
5	Робоче місце при виконанні робіт сидячи.	Робоче місце	ДЕСТ 12.2.032-78

5.2 Інженерно-технічні заходи з охорони праці

5.2.1 Заходи щодо забезпечення електробезпеки

Відповідно до класифікації ПУЕ за небезпекою ураження електричним струмом приміщення операторів ПК відноситься до приміщень з підвищеною небезпекою, так як існує можливість одночасного дотику до опалювальних батарей приміщення, з'єднаних з землею і корпусів електрообладнання. В операторській використовується обладнання з напругою живлення 220 В.

Лінія електромережі для живлення ПК та периферійного обладнання виконується як окрема групова трьохпровідна мережу шляхом прокладки фазного і нульового робочого та захисного провідників. Нульовий захисний провідник служить для занулення електроприймачів. При напрузі до 1000 В застосовують трьохпровідну мережу з ізольованою нейтраллю.

Електропроводка в операторській повинна бути виконана прихованим методом, прокладена в гнучких металоруковах, що робить силові ланцюги недоступними для працюючих.

Струмовий захист реалізується з використанням автоматів, які розривають електричну мережу при високих струмах навантаження. Для забезпечення захисного відключення використовуємо УЗО ВД1-63, основним призначенням якого є забезпечення безпеки людини в разі дотику до занулення (заземленому) корпусу при замиканні на нього фази, а також при безпосередньому дотику до струмоведучих частини електроустановки.

Основні заходи, спрямовані на попередження випадків ураження електричним струмом в операторській, такі:

− щодня проводити очистку монітора від пилу;

− забороняється знімати захисну кришку системного блоку комп'ютера;

– усунення можливості випадкового дотику до струмоведучих частин електроустаткування, що знаходиться під напругою;
– малій напруги;
– надійна ізоляція струмоведучих частин електрообладнання і своєчасний його ремонт;
– захисне занулення;
– захисне відключення та застосування плавких запобіжників.

5.2.2 Заходи щодо захисту від підвищеної температури

При виконанні робіт, пов'язаних з нервово-емоційним напруженням у приміщені з робочим місцем розробника повинні дотримуватися оптимальні умови мікроклімату (температура повітря 22 - 24 °С, відносна вологість 60 - 40%, швидкість руху повітря не більше 0,1 м/сек.).

Кімната повинна бути обладнана власною системою вентиляції та кондиціонування на базі спліт-системи LG A12LHR продуктивністю 510 $м^3$/год, яка стабілізує температуру повітря в регульованих межах 14 ... 32 ± 2°С.

5.2.3 Освітлення робочої зони

В приміщенні з робочим місцем використовується поєднане природне і штучне освітлення. Згідно СНиП 23.05- 95 – середня точність зорової роботи, найменший розмір об'єкта розрізнення складає 0,3÷0,5 мм.

Штучне освітлення може бути двох систем – загальне і комбіноване.
Для освітлення приміщення використані, найбільш економічні люмінесцентні лампи типу ЛБ. Для місцевого освітлення використані лампи розжарювання.
На робочому місці відсутні різкі тіні.

Для внутрішнього оздоблення інтер'єру приміщень з ПК використані диффузионно-відбивні матеріали з коефіцієнтами відбиття світла для стелі 0,7÷0,8; для стін 0,5÷0,6; для підлоги 0,3÷0,5.

5.2.4 Заходи по боротьбі з шкідливими факторами при роботі з персональним комп'ютером

Розробники за характером роботи багато часу проводять з ПК. В даному випадку працівник схильний до впливу напруги зору. Це призводить до підвищеного стомлення зору і загального стомлення.

Для безпечної і комфортної роботи при експлуатації персонального комп'ютера розроблені наступні заходи:

1. Площа на одне робоче місце має становити не менше 6 м^2, об'єм – не менше 20 м^3.
2. Висота робочої поверхні для монітора повинна становити 680-800 мм (рекомендовані розміри столу: висота - 725 мм, ширина - 600-1400 мм, глибина - 800-1000 мм).
3. Робоче сидіння працівника має складатися з: сидіння, спинки і підлокітників.
4. Монітор і клавіатура повинні бути розміщені на поверхні столу або на спеціальній робочій поверхні окремо від столу, яка повинна бути відрегульована по висоті, на відстані 100-300мм від краю.
5. Щоб освітлення не створювало сліпучих відблисків, комп'ютер повинен бути розташований так, щоб пряме світло не попадало на екран.
6. Верхній край екрана слід розташовувати на рівні очей або трохи нижче.
7. Оптимальна відстань від очей до екрана 600-700 мм.

8. Покриття підлоги повинне бути матовим з коефіцієнтом відбиття 0,3-0,5;

9. Вологе прибирання повинне проводитися на початку робочого дня, а також під час перерви.

5.3 Заходи з ергономіки

Проектування робочого місця оператора ПК має супроводжуватися прагненням поліпшити обстановку, яка буде сприяти збереженню високої працездатності, і створювати сприятливі умови для співпраці працівників. Стандарт повинен обумовлювати те, що робоче місце і взаємне розташування всіх його елементів повинне відповідати антропометричним, фізичним і психологічним вимогам.

Проектування робочих місць, забезпечених моніторами, відноситься до числа важливих проблем ергономічного проектування в області обчислювальної техніки. При конструюванні робочих місць необхідно отримуватися наступні основні умови:

а) достатній робочий простір для оператора, що дозволяє здійснювати всі необхідні рухи і переміщення при експлуатації та технічному обслуговуванні обладнання;

б) достатні фізичні, зорові і слухові зв'язки між операторами та обладнанням, а також між операторами;

в) оптимальне розміщення робочих місць у приміщеннях для оперативної роботи, а також безпечні і достатні проходи для операторів;

г) оптимальне розміщення устаткування (головним чином засобів відображення інформації й органів керування), завдяки чому забезпечується зручне положення оператора при роботі;

д) чітке позначення органів керування, індикаторів та інших елементів обладнання, які потрібно знаходити, ідентифікувати і якими доводиться маніпулювати (маркування не є обов'язковим для органів управління або обладнання, призначення яких очевидно для оператора);

е) необхідне природне і штучне освітлення для виконання оперативних завдань,

технічного обслуговування" або тренувань;

ж) припустимий рівень акустичного шуму і вібрації, створюваних устаткуванням робочого місця або іншими джерелами шуму і вібрації;

з) достатню простоту і швидкість збірки і розбирання устаткування;

и) виключення можливості неправильної установки, заміни та монтажу блоків або елементів обладнання, а також неправильної ідентифікації, орієнтації і розташування кабелів і роз'ємів;

к) наявність необхідних інструкцій і попереджувальних знаків, застережливих про небезпеки, які можуть виникнути при роботі, і вказують на необхідні заходи безпеки;

л) необхідні опори і підставки для тимчасового розміщення вийнятих блоків або елементів обладнання, а також для випробувального обладнання, приладів, інструментів і технічних посібників;

м) надійну індикацію для випадків відмови електричного живлення, а також відмови обладнання або його функціонування з виходом за допустимі межі.

Згідно з ГОСТ 12.2.032-78 висота робочої поверхні при організації робочого місця повинна бути:

а) Для жінок – 630 мм.

б) Для чоловіків 680 мм.

Висота сидіння:

а) Для жінок – 400 мм.

б) Для чоловіків 430 мм.

Підставка для ніг повинна бути регульованою по висоті. Ширина повинна бути не менше 300 мм, довжина - не менше 400 мм. Поверхня підставки повинна бути рифленою. По передньому краю слід передбачати бортик висотою 10 мм

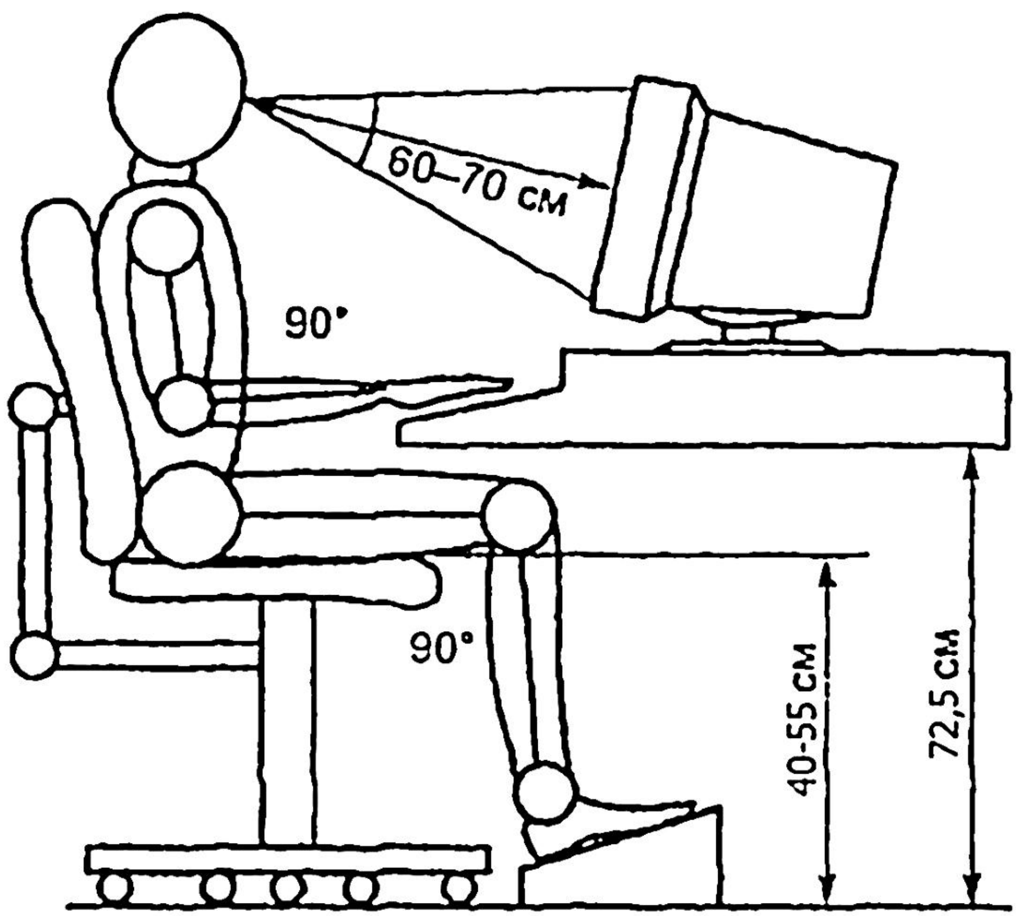

Рисунок 5.1 – Робоче місце оператора та його положення

ВИСНОВОК

В рамках виконання кваліфікаційної роботи бакалавра було досліджено існуючі на ринку системи управління персоналом, проаналізовані сучасні веб-технології с застосуванням яких розробляються такі системи та формалізовано вимоги до розроблюваної системи. Основні критерії дослідження – це функціонал систем, на скільки вони підходять малому бізнесу та цінова політика ліцензій на використання системи. На підставі проведених досліджень було виявлено, що основний фактор який унеможливлює використання досліджених систем малими підприємствами це необхідність налаштування перед використанням, навчання співробітників та завелика ціна обумовлена наявністю функціоналу необхідного лише великим підприємствам.

Під час аналізу сучасних веб технологій було виявлено що наразі найбільш гнучким набором технологій для розробки інтерфейсів веб-застосунків будь якого масштабу є HTML, CSS та JavaScript. З використанням цих основних інструментів, ряду допоміжних бібліотек та фреймворків можна створювати додатки різних рівнів складності.

Для комунікації інтерфейсу с веб-серверами найкращім вибором є протокол HTTP в якому наявні сучасні алгоритми стиснення та шифрування.

Для збереження даних та хостингу веб-застосунку для додатків будь-якої складності найкращим вибором є сервіс AWS. Оскільки він має підтримку майже нескінченного масштабування та характеризується виключною відмовостійкістю.

ПЕРЕЛІК ПОСИЛАНЬ

1. Дипломування. Методичні вказівки для спеціальності 123 Комп’ютерна інженерія / Л.І. Цвіркун, С.М. Ткаченко, Я.В. Панферова, Д.О. Бешта, Л.В. Бешта. – Д.: НТУ «Дніпровська політехніка», 2020. – 69 с.

2. Методичні вказівки до розділу «Охорона праці» в дипломних роботах бакалаврів / В.І. Голінько, В.Ю. Фрундін, Ю.І. Чеберячко, М.Ю. Іконніков. – Д.: Державний ВНЗ «Національний гірничий університет», 2012. – 8 с.

3. Alex Banks, Eve Porcello, O’Reilly Learning React: Functional Web Development with React and Redux 1st Edition, 2017. – 625 с.

4. Design Patterns: Elements of Reusable Object-Oriented Software / E.Gamma, J. Vlissides, R. Johnson, R. Helm, 1994. – 395 с.

5. Tim Ambler JavaScript Frameworks for Modern Web Dev, 2015. – 520 с.

6. Mark Masse, REST API Design Rulebook 1st Edition, 2011. – 114 с.

7. Micah Godbolt, O’Reilly Frontend Architecture for Design Systems: A Modern Blueprint for Scalable and Sustainable Websites, 2016. – 198 с.

8. Robin Wieruch, The Road to React: Your journey to master 2017. – 228 с.

9. AWS Cognito [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/ru/cognito

10. Thinking in React [Електронний ресурс] – Режим доступу до ресурсу: https://reactjs.org/docs/thinking-in-react.html#step-5-add-inverse-data-flow

11. AWS [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/

12. The official Semantic-UI-React integration [Електронний ресурс] – Режим доступу до ресурсу: https://react.semantic-ui.com/

13. React vs. angular [Електронний ресурс] – Режим доступу до ресурсу: https://www.educative.io/edpresso/react-vs-angular

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
“ДНІПРОВСЬКА ПОЛІТЕХНІКА”

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ
ЛІСТИНГ ПРОГРАММИ

Текст програми

804.02070743.20005-01 12 09

Листів 21

2020

АНОТАЦІЯ

Даний лістинг містить в собі частину програмного коду для програмування роботи клієнтської частини застосунку. Показані основні підходи до роботи с браузерними сховищам, перехват запитів, маршрутизація запитів на сторінки, робота з історією, робота з компонентом контролю взаємодії з даними про автентифікацію що знаходяться у сховищі, ініціалізація константних значень, та файли конфігурацій.

ЗМІСТ

1. Створення точки входу застосунку

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <link href="https://fonts.googleapis.com/css?family=Roboto:500,700&display=swap"
rel="stylesheet">
    <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1,
user-scalable=no">
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Staff Management software"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <title>Staff Management</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
// ініціалізація головного компоненту сторінки
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import 'semantic-ui-css/semantic.min.css';

ReactDOM.render(
  <React.StrictMode>
```

```
    <App />
  </React.StrictMode>,
  document.getElementById('root'),
);
```

2. Налаштування запитів до базового API

```
import axios from 'axios';
import { responseInterceptor, errorInterceptor, requestInterceptor } from './interceptors';

const baseURL = process.env.REACT_APP_API_URL;

const baseAPI = axios.create({
  baseURL,
  headers: {
    'Content-type': 'application/json; charset=UTF-8',
  },
  withCredentials: true,
});

baseAPI.interceptors.response.use(
  (request) => requestInterceptor(request),
);
baseAPI.interceptors.response.use(
  (response) => responseInterceptor(response),
  (error) => errorInterceptor(error),
);

export default baseAPI;
```

3. Налаштування перехвату запитів

```
import { AUTHORIZATION } from 'api/constants';
import storage from './storage';
import history from '../history';
```

```
const handleSignIn = (response) => {
  const NEW_PASSWORD_REQUIRED = 'New password is required';

  if (response.data?.message === NEW_PASSWORD_REQUIRED) {
    const userEmail = response.config.data && JSON.parse(response.config.data).email;
    storage.set('authorization', AUTHORIZATION.PASSWORD_CHANGE);
    storage.set('email', userEmail);
  } else if (response.data?.userID) {
    storage.set('userId', response.data?.userID);
    storage.set('email', '');
    storage.set('authorization', AUTHORIZATION.AUTHORIZED);
  }

  return response;
};

const handleSignUp = (response) => {
  const isAuthorized = storage.get('authorization') == AUTHORIZATION.AUTHORIZED;

  if (!response.data?.message) {
    if (isAuthorized) {
      history.push('/');
    } else {
      history.push('/login');
    }
  }

  return response;
};

const handleReset = (response) => {
  if (!response.data?.message) {
```

```
    storage.set('authorization', AUTHORIZATION.UNAUTHORIZED);
    history.push('/login');
  }

  return response;
};

const handleUser = (response) => {
  const role = response.data?.role;

  if (role) {
    storage.set('role', role);
  }

  return response;
};

const handleSignOut = () => {
  storage.set('authorization', AUTHORIZATION.UNAUTHORIZED);
  storage.set('role', '');
  storage.set('email', '');
  storage.set('userId', '');
  history.push('/login');
};

export const responseInterceptor = (response) => {
  if (response.status === 401) {
    handleSignOut();
    return response;
  }

  switch (response.config.url) {
    case '/signin': return handleSignIn(response);
```

```
    case '/signup': return handleSignUp(response);
    case '/user': return handleUser(response);
    case '/password/required': return handleReset(response);

    default:
      return response;
  }
};

export const requestInterceptor = (request) => request;

export const errorInterceptor = (error) => {
  const customMessage = error.response?.data.message;

  if (error.response?.status === 401) {
    handleSignOut();
  }

  if (error.response?.status === 500) {
    alert(`Request ${error.response.config.url} failed due to server error, page will be reloaded. If
error persists contact administrator`);
    window.location.reload();
  }

  return Promise.reject(
    customMessage ? new Error(customMessage) : error,
  );
};
```

4. Маршрутизація запитів на сторінки

```
import React, {
  lazy,
  Suspense,
```

```
  useState,
  useEffect,
} from 'react';
import {
  Route,
  Switch,
  Router,
  Redirect,
} from 'react-router-dom';
import storage from './api/storage';
import Layout from './layout';
import SignUp from './pages/signup';
import { AUTHORIZATION } from './api/constants';
import history from './history';

import Home from './pages/home';
import Task from './pages/task';
import Login from './pages/login';
import Reset from './pages/reset';
import SignOut from './pages/sign-out';
import Vacations from './pages/vacations';
/*
const Task = lazy(() => import('./pages/task'));
const Home = lazy(() => import('./pages/home'));
const Login = lazy(() => import('./pages/login'));
const SignOut = lazy(() => import('./pages/sign-out'));
const Reset = lazy(() => import('./pages/reset'));
*/

const App = () => {
  const [authorization, setAuthorization] = useState(storage.get('authorization'));

  useEffect(() => {
```

```
  const authListener = () => {
    const auth = storage.get('authorization');

    if (Object.values(AUTHORIZATION).includes(auth)) {
      setAuthorization(auth);
    } else {
      storage.set('authorization', AUTHORIZATION.UNAUTHORIZED);
      setAuthorization(AUTHORIZATION.UNAUTHORIZED);
    }
  };

  authListener();

  return storage.subscribe(authListener);
}, []);

return (
  <Router history={history}>
    <Layout>
      <Suspense fallback={<div>Loading...</div>}>
        {
          authorization === AUTHORIZATION.AUTHORIZED && (
          <Switch>
            <Route path="/" exact component={Home} />
            <Route path="/create" component={Home} />
            <Route path="/user/:id" exact component={Home} />
            <Route path="/task/:id" component={Task} />
            <Route path="/vacations/:id" component={Vacations} />
            <Route path="/vacations/" component={Vacations} />
            <Route path="/adduser" exact component={SignUp} />
            <Route path="/signout" exact component={SignOut} />
            <Route path="/reset" exact component={Reset} />
            <Redirect to="/" />
```

```
          </Switch>
          )
        }
        {
          authorization === AUTHORIZATION.PASSWORD_CHANGE && (
          <Switch>
            <Route path="/reset" exact component={Reset} />
            <Redirect to="/reset" />
          </Switch>
          )
        }
        {
          authorization === AUTHORIZATION.UNAUTHORIZED && (
          <Switch>
            <Route path="/adduser" exact component={SignUp} />
            <Route path="/login" exact component={Login} />
            <Redirect to="/login" />
          </Switch>
          )
        }
      </Suspense>
    </Layout>
  </Router>
 );
};

export default App;
```

5. Робота із локальним сховищем

```
const storage = {
 get(key, defaultValue) {
  const value = window.localStorage.getItem(key);
```

```
    return value === null ? defaultValue : value;
  },
  set(key, value) {
    try {
      window.localStorage.setItem(key, value);
      window.dispatchEvent(new Event('storage:change'));
    } catch (error) {
      console.error('Can\'t set item to storage:', key, value);
    }
  },
  remove(key) {
    window.localStorage.removeItem(key);
    window.dispatchEvent(new Event('storage:change'));
  },
  subscribe(listener) {
    window.addEventListener('storage:change', listener);

    return () => window.removeEventListener('storage:change', listener);
  },
};

export default storage;
```

6. Компонент контролю автентифікації

```
import React, { useEffect, useState } from 'react';
import { AUTHORIZATION, USER_ROLE } from 'api/constants';
import storage from 'api/storage';

const withAuth = (WrappedComponent) => (props) => {
  const [userId, setUserId] = useState(storage.get('userId'));
  const [role, setRole] = useState(storage.get('role'));
  const [isAuthorized, setIsAuthorized] = useState(storage.get('authorization') ===
AUTHORIZATION.AUTHORIZED);
```

```
useEffect(() => {
  const authListener = () => {
    const storedAuth = storage.get('authorization');
    const storedRole = storage.get('role');
    const storedUserId = storage.get('userId');

    setUserId(storedUserId);

    if (Object.values(USER_ROLE).includes(storedRole)) {
      setRole(storedRole);
    }

    if (storedAuth === AUTHORIZATION.AUTHORIZED) {
      setIsAuthorized(true);
    } else {
      setIsAuthorized(false);
    }
  };

  authListener();

  return storage.subscribe(authListener);
}, []);

return (
  <WrappedComponent
    // eslint-disable-next-line react/jsx-props-no-spreading
    {...props}
    authorizedUserId={userId}
    userRole={role}
    isAdmin={role === USER_ROLE.ADMIN}
    isAuthorized={isAuthorized} />
```

```
  );
};

export default withAuth;
```

7. Ініціалізація базових константних значень

```
import { USER_ROLE } from 'api/constants';

export const ROLES = [
  { key: USER_ROLE.USER, value: USER_ROLE.USER, text: 'User' },
  { key: USER_ROLE.ADMIN, value: USER_ROLE.ADMIN, text: 'Manager' },
];

export const TASK_STATE = {
  IN_PROGRESS: 'InProgress',
  READY_FOR_DEV: 'Ready',
  BLOCKED: 'Blocked',
  DONE: 'Done',
};

export const TASK_STATE_STRING = {
  InProgress: 'In progress',
  Ready: 'Ready for dev',
  Blocked: 'Blocked',
  Done: 'Done',
};

export const TASK_STATE_OPTIONS = [
  { key: TASK_STATE.IN_PROGRESS, value: TASK_STATE.IN_PROGRESS, text:
TASK_STATE_STRING[TASK_STATE.IN_PROGRESS] },
  { key: TASK_STATE.READY_FOR_DEV, value: TASK_STATE.READY_FOR_DEV, text:
TASK_STATE_STRING[TASK_STATE.READY_FOR_DEV] },
```

```
  { key: TASK_STATE.DONE, value: TASK_STATE.DONE, text:
TASK_STATE_STRING[TASK_STATE.DONE] },
  { key: TASK_STATE.BLOCKED, value: TASK_STATE.BLOCKED, text:
TASK_STATE_STRING[TASK_STATE.BLOCKED] },
];

export const VACATION = {
  PENDING: 'Pending',
  APPROVED: 'Approved',
  REJECTED: 'Rejected',
  CANCELED: 'Canceled',
  EXPIRED: 'Expired',
};

export const USER_ROLE = {
  ADMIN: 'admin',
  USER: 'user',
};

export const AUTHORIZATION = {
  AUTHORIZED: 'AUTHORIZED',
  PASSWORD_CHANGE: 'PASSWORD_CHANGE',
  UNAUTHORIZED: 'UNAUTHORIZED',
};
```

8. Налаштування статичного аналізу коду

```
{
  "settings": {
    "import/resolver": {
      "node": {
        "paths": ["src"]
      }
    }
```

```
 },
 "extends": [
  "react-app",
  "airbnb",
  "plugin:jsx-a11y/recommended"
 ],
 "plugins": [
  "jsx-a11y"
 ],
 "rules": {
  "react/jsx-filename-extension": [1, { "extensions": [".js", ".jsx"] }],
  "react/forbid-prop-types": "off",
  "react/destructuring-assignment": "off",
  "react/state-in-constructor": "off",
  "jsx-a11y/click-events-have-key-events": "off",
  "jsx-a11y/no-static-element-interactions": "off",
  "jsx-a11y/label-has-associated-control": "off",
  "import/prefer-default-export": "off",
  "react/jsx-closing-bracket-location": [1, "after-props"],
  "max-len": ["error", { "code": 120 }]
 }
}
```

9. Конфігурація метаданих застосунку

```
{
 "name": "STFMGMT",
 "version": "1.0.0",
 "private": true,
 "dependencies": {
  "@testing-library/jest-dom": "^4.2.4",
  "@testing-library/react": "^9.3.2",
  "@testing-library/user-event": "^7.1.2",
  "axios": "^0.19.2",
```

```
  "classnames": "^2.2.6",
  "debounce": "^1.2.0",
  "husky": "^4.2.5",
  "moment": "^2.25.3",
  "prop-types": "^15.7.2",
  "pure-react-carousel": "^1.27.0",
  "react": "^16.13.1",
  "react-dates": "^21.8.0",
  "react-dom": "^16.13.1",
  "react-dropzone": "^11.0.1",
  "react-image-crop": "^8.6.2",
  "react-input-autosize": "^2.2.2",
  "react-router-dom": "^5.1.2",
  "react-scripts": "3.4.1",
  "semantic-ui-css": "^2.4.1",
  "semantic-ui-react": "^0.88.2"
},
"scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject",
  "lint": "eslint ."
},
"browserslist": {
  "production": [
    ">0.2%",
    "not dead",
    "not op_mini all"
  ],
  "development": [
    "last 1 chrome version",
    "last 1 firefox version",
```

```
    "last 1 safari version"
   ]
  },
  "devDependencies": {
   "eslint-config-airbnb": "^18.1.0",
   "eslint-config-prettier": "^6.10.1",
   "eslint-plugin-jsx-a11y": "^6.2.3",
   "eslint-plugin-prettier": "^3.1.3"
  },
  "husky": {
   "hooks": {
    "pre-commit": "yarn build"
   }
  }
}
```

10. Налаштування маршрутизації діалогових вікон

```
import React from 'react';
import {
  Switch,
  Route,
} from 'react-router-dom';
import TaskCreateModal from 'components/task-create-modal';
import VacationRequestModal from 'components/vacation-request-modal';
import history from '../../history';

const Modals = () => {
  const closeTaskModal = () => {
    history.push('/');
  };

  return (
    <Switch>
```

```
      <Route path="/create/task" component={() => <TaskCreateModal
onClose={closeTaskModal} />} />
      <Route path="/create/vacation" component={() => <VacationRequestModal
onClose={closeTaskModal} />} />
    </Switch>
  );
};

export default Modals;
```

11. Метод обрізки зображень за допомогою canvas API

```
const getCroppedImg = (image, crop, blob = false) => {
  if (!crop.width) return null;
  const canvas = document.createElement('canvas');
  const scaleX = image.naturalWidth / image.width;
  const scaleY = image.naturalHeight / image.height;
  canvas.width = crop.width;
  canvas.height = crop.height;
  const ctx = canvas.getContext('2d');

  ctx.drawImage(
    image,
    crop.x * scaleX,
    crop.y * scaleY,
    crop.width * scaleX,
    crop.height * scaleY,
    0,
    0,
    crop.width,
    crop.height,
  );

  if (blob) {
```

```
    const getCanvasBlob = () => new Promise((resolve) => {
      canvas.toBlob((canvasBlob) => resolve(canvasBlob));
    });

    return getCanvasBlob();
  }

  return canvas.toDataURL('image/jpeg', 1);
};

export default getCroppedImg;
```

12. Запит на інвалідацію токену

```
import React, { useEffect } from 'react';
import storage from 'api/storage';
import baseAPI from 'api/config';
import { AUTHORIZATION } from 'api/constants';

const SignOut = () => {
  useEffect(() => {
    const signOut = async () => {
      await baseAPI.post('/signout');
      storage.set('authorization', AUTHORIZATION.UNAUTHORIZED);
      storage.set('role', '');
      storage.set('email', '');
      storage.set('userId', '');
    };
    signOut();
  }, []);

  return 'Signing out';
};
```

```
export default SignOut;
```

13. Конфігурація системи контролю версій

```
# dependencies
/node_modules
/.pnp
.pnp.js

# testing
/coverage

# production
/build

# misc
.DS_Store
.idea
.env
.env.local
.env.development.local
.env.test.local
.env.production.local

npm-debug.log*
yarn-debug.log*
yarn-error.log*
```