МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

першого (бакалаврського) рівня вищої освіти

на тему: **«Розробка інформаційної системи обліку робочого часу у відділі комп'ютерних інформаційних технологій Львівського національного університету природокористування»**

Виконав: здобувач освіти групи Іт-41
Спеціальності 126 «Інформаційні системи та технології»
(шифр і назва)

Бунга Володимир Романович
(Прізвище та ініціали)

Керівник: к.е.н. Станько В.Ю.
(Прізвище та ініціали)

Рецензент: к.т.н. Березовецький С.А.
(Прізвище та ініціали)

ДУБЛЯНИ-2024

УДК 004:331.31] :378.4(477.83)

Розробка інформаційної системи обліку робочого часу у відділі КІТ. Бунга В.Р. – Кваліфікаційна робота бакалавра. Кафедра інформаційних технологій – Дубляни, ЛНУП, 2024.

49 сторінок текстової частини, 15 рисунків, 1 таблиця та 15 джерел інформації.

Текстова частина включає вступ, чотири розділи, висновки, список використаних джерел.

У вступі висвітлено важливість інформаційних систем обліку робочого часу. Підкреслено, яку роль відіграють інформаційні системи обліку робочого часу в сучасному світі .

У першому розділі подано аналіз предметної області, де описується розвиток інформаційних систем та технологій. Описано сучасний стан інформаційних систем обліку робочого часу.

У другому розділі описано мови програмування, їх різновидність в веб-розробці. Обгрунтовано вибір технологій для розробки інформаційної системи.

У третьому розділі розроблено Use case діаграму, блок-схему інформаційної системи. Створено клієнтську частину інформаційної систему. Створено серверну частину інформаційної системи. Описано вигляд сторінок системи, інтеграція з Google Sheet API, обробка даних та хешування паролів.

У четвертому розділі розроблені заходи з охорони праці і навколишнього середовища.

На підставі виконаної роботи зроблено відповідні висновки.

ЗМІСТ

ВСТУП

У сучасних реаліях, коли управління ресурсами та робочими процесами є головними факторами для успішної діяльності будь-якої організацій, облік робочого часу стає необхідністю для забезпечення нормального функціонування персоналу. Відділ комп'ютерно-інформаційних технологій (КІТ) Львівського національного університету природокористування, як один з основних елементів управління та технічної підтримки інформаційної інфраструктури університету, потребує надійної системи обліку робочого часу для персоналу.

Мета даної дипломної роботи полягає в розробці інформаційної системи, яка забезпечить надійний та автоматизований облік робочого часу працівників відділу КІТ ЛНУП. Враховуючи основні особливості та потреби відділу, завданням дипломної роботи буде оцінка існуючих систем обліку робочого часу та розробка ефективної інформаційної системи з усіма необхідними складовими.

Об'єктом дослідження в даній роботі є облік робочого часу працівників у відділі КІТ ЛНУП, а предметом дослідження - вирішення проблем, пов'язаних із точністю та зручністю інформаційних систем обліку робочого часу. Виняткова увага буде надана аналізу існуючих процесів обліку робочого часу, визначення їх недоліків та потреб удосконалення.

Розроблена інформаційна система, яка дозволить відділу КІТ контролювати робочий час, забезпечить надійний облік даних. Очікується, що впровадження цієї системи значно покращить управління ресурсами відділу, знизить витрати часу на адміністративні процедури та сприятиме загальній ефективності роботи університету. Крім того, розроблена інформаційна система буде здатна адаптуватися до зростаючих потреб та вимог, що виникають у зв'язку з розвитком відділу КІТ та змінами університетських процесів.

Планується, що розроблена інформаційна система стане важливим кроком у модернізації управління персоналом та оптимізації робочих процесів університету в цілому. Очікується, що її впровадження дозволить підвищити ефективність роботи відділу КІТ та покращити загальний стандарт обслуговування співробітників. Результати цієї дипломної роботи відкриють нові можливості для подальшого розвитку систем управління трудовими ресурсами та діджиталізації у Львівському національному університеті природокористування.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Аналіз предметної області для системи обліку робочого часу передбачає детальне дослідження та оцінку інших поширених методів обліку, а також аналіз вимог майбутніх користувачів.

Сфера обліку робочого часу за останні декілька років зазнала великих змін, рухаючись в сторону автоматизації та оцифрування. Замість ручного ведення журналів і підрахунку робочих днів, більшість підприємств почали використовувати спеціалізовані програми та інформаційні системи, які спрощують та оптимізують процес обліку часу. Це все надає наступні переваги:

- Підвищення точності. Такі системи автоматично передають час і дату прибуття на робоче місце. Виключають помилки пов’язані з людським фактором.
- Економія часу. За допомогою систем обліку робочого часу, керівник відділу та працівники можуть використовувати зекономлений час на більш продуктивні речі.
- Прозорість. Системи обліку робочого часу висвітлюють чітко відпрацьовані дні кожним працівником, що полегшує обрахунок заробітної плати.

Впровадження систем обліку робочого часу стають актуальнішими кожного дня для організацій та підприємств будь-яких масштабів та сфери діяльності. Це дозволяє таким підприємствам та організаціям оптимізувати управління колективом, підвищити ефективність та економити кошти

В даній дипломній роботі буде розроблена інформаційна система обліку робочого часу. Ця система буде спеціалізуватись на веденні таблиці робочого часу кожного працівника відділу КІТ Львівського національного університету природокористування.

1.1 Сучасний рівень обліку робочого часу працівників

Розвиток інформаційних систем та технологій спростив великий спектр процесів. Важливою складовою даного розвитку є використання обліку робочого часу, яке сильно спрощує роботу підприємств. Адже це нововведення допомагає підвищити продуктивність праці на підприємствах.

Надалі розвиток інформаційних систем та технологій в обліку робочого часу працівників буде досягати нових висот і перебіг праці буде комфортний та більшою мірою модернізований. Це не лише може стати в пригоді для точного обліку часу відвідуваності робочого місця, а й надавати ефективний моніторинг відпусток, лікарняних та прогулів тому, що ці показники мають вирішальне значення для нарахування заробітної плати, нарахування податків та планування робочого навантаження.

Протягом тривалого часу підприємства використовували різні методи обліку робочого часу працівників. Найпопулярнішим серед багатьох інших був паперовий облік, коли працівники вручну заповнювали всі відомості про робочий час у табелі, або інші паперові документи. Цей метод був і є простим і зрозумілим для більшості людей, але він мав декілька вагомих недоліків, такі як низька точність, ризик помилок при внесенні даних в табель, складність обробки та зберігання великої кількості документації.

В Україні зауважується тенденція до впровадження сучасних інформаційних систем обліку робочого часу, особливо, беручи до уваги популярні та нестандартні гнучкі графіки роботи та віддалений формат роботи. Потрібно також взяти до уваги, що існують деякі труднощів, такі як консерватизм керівників, слабка ІТ-грамотність більшості працівників та ризики пов’язані з конфіденційністю даних. Для вирішення цих труднощів потрібно докласти чимало зусиль, і також інвестицій для навчання працівників та їх адаптації. Підприємства, які використовують сучасні інформаційні системи обліку робочого часу, отримують багато переваг, це робить їх біль конкурентоспроможними на ринку праці.

1.2 Аналіз сфери існуючих систем

На даний момент існує багато систем обліку робочого часу працівників, як на великих, так і на зовсім малих підприємствах. В основному державні структури в Україні використовують традиційні паперові системи обліку робочого часу працівників, а більшість приватних організацій використовують сучасні методи, які полегшують роботу кожного в незалежності від кількості працівників, розміру компанії. Розглянемо основні типи таких систем:

Традиційні системи обліку робочого часу. Такі системи обліку робочого часу є найстарішими та найбільш поширеними. Вони відстежують кількість витраченого часу на виконання обов'язків працівника. Традиційні системи обліку робочого часу включають в себе ручне ведення записів у вигляді щоденників або табелів, а також систематичну звітність про витрачений час працівника. Недоліками таких систем є людський фактор, тобто людина може допустити помилку при заповненні документів, доступність таких документів, крім того ручне оброблення даних є досить трудомістким, що вимагає значних зусиль.

Автоматизовані системи обліку робочого часу працівників. Ці сучасні системи обліку робочого часу дозволяють досить сильно полегшити процес відстеження відпрацьованого часу співробітником. Основними типами автоматизованих систем обліку робочого часу є: карти-перепустки, біометричні дані, мобільні додатки, GPS-трекери, time tracker. Вони допомагають точно оцінювати відпрацьовані години, понаднормові години, запізнення.

- Системи на основі біометричної ідентифікації. Хотілось би виділити ці системи тому, що зараз вони здобувають велику популярність серед сучасних компаній. Ці системи використовують фізичні характеристики кожного працівника, такі як відбитки пальців, малюнок райдужної оболонки ока, або сканування обличчя для обліку робочого часу. Біометричні системи забезпечують вискоий рівень безпеки, підробки даних.

- Мобільні додатки. Розвиток мобільних технологій приводить нас до появи різноманітних додатків, в тому числі додатків відстеження робочого часу працівників. Такі додатки працюють на більшості сучасних смартфонах і планшетах, і дозволяють працівникам фіксувати початок і кінець робочого дня. Використання мобільних додатків для обліку робочого часу працівників є зручними і для самих працівників.

Таблиця 1.1 – Таблиця порівняння систем обліку робочого часу

Критерії	**Традиційні паперові системи**	**Найпростіша автоматизована система**	**Біометрична системи**	**Мобільні додатки**
Точність обліку	Низька	Висока	Найвища	Висока
Зручність	Низька	Висока	Середня	Середня
Захищеність даних	Низька	Висока	Найвища	Висока
Вартість	Низька	Середня	Висока	Середня

Порівняльна таблиця 1.1 дозволяє визначити сильні та слабкі сторони кожної системи. Потрібно взяти до уваги, що такі системи вимагають початкових інвестицій в IT-інфраструктуру підприємства, також вони можуть викликати опір, або недовіру серед працівників. Тому при впровадженні потрібно враховувати не тільки технічні, а й організаційні фактори. Як традиційні, так і автоматизовані системи обліку мають свої переваги та недоліки, але беззаперечним є те, що перехід від застарілих паперових методів дуже спрощує процес обліку робочого часу працівників.

1.3 Нормативно-правові вимоги

Облік робочого часу працівників є важливою частиною роботи підприємства та дотримання трудового законодавства України. Сучасні інформаційні системи обліку робочого часу повинні відповідати ряду нормативно-правових вимог для забезпечення законності та захисту персональних даних кожного з працівників. Такі вимоги керуються трудовим кодексом, законодавством про захист персональних даних.

Інформаційні системи обліку робочого часу працівників повинні повністю відповідати вимогам чинного трудового законодавства України. Це означає, що вони мають забезпечувати точний облік та фіксацію відпрацьованого часу, дотримання норм тривалості робочого часу, оплати праці, надання відпусток, оплата понаднормових годин та інших встановлених законодавством України гарантій для працівників, незалежно від розміру та кількості працівників в організації. Також такі системи повинні відповідати вимогам Законів України «Про охорону праці», «Про відпустки», «Оплату праці» та інших нормативно-правових актів, які стосуються кодексу законів «Про працю в Україні». Найбільша увага має приділятись автоматизованому обліку часу та контролю за дотриманням норм тривалості робочого часу, нічних змін, надурочних робіт та щорічних відпусток. Це все потрібно для забезпечення законних прав кожного з працівників та уникнення скарг з боку державних органів.

Захист персональних даних працівників є одним з основних вимог законодавства. Роботодавці зобов’язані забезпечувати безпеку та конфіденційність персональних даних всіх працівників. Сюди входить збір, зберігання, обробка та передача даних про працівників тільки з їхньої згоди та у встановлених межах. Працівники мають повне право знати, які дані обробляються, з якою метою та вимагати виправлення чи видалення недостовірних даних. Порушення вимог щодо захисту персональних даних може призвести до юридичних та фінансових наслідків для підприємства.

Забезпечення конфіденційності та безпеки даних співробітників є однією з основних вимог. Захист персональних даних та конфіденційність інформації, що міститься в системі обліку робочого часу, є першорядною вимогою до інформаційної системи. Потрібно запровадити заходи для ефективного забезпечення безпеки даних, включаючи :

- Контроль доступу – встановлення правил доступу до системи та даних, з обмеженням доступу тільки для уповноважених користувачів.
- Шифрування даних – використати методи для забезпечення конфіденційності інформаціїї.
- Резервне копіювання – систематичне створення резервних копій для запобігання їх втрати.

Дотримання вимог дійсного законодавства України з питань захисту персональних даних є обов’язковим.

РОЗДІЛ 2
ВИБІР БАЗОВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У цьому розділі буде розглянуто питання вибору базового програмного забезпечення для розробки інформаційної системи обліку робочого часу працівників. Вибір правильного програмного забезпечення є основним для успішної розробки інформаційної системи. Основні технології, які ми використали включають: мова розмітки – HTML, мова програмування – JavaScript, мова програмування – Python, фреймворк – Flask, мова програмування взаємодії користувача з базами даних – SQL, мова стилю сторінок – CSS. Взаємодія цих технологій створює інформаційну систему, здатну ефективно вирішувати завдання обліку робочого часу. Кожна з цих технологій відіграє важливу роль у створенні інформаційної системи.

2.1 Аналіз популярних мов програмування для веб-розробки

Веб розробка - це процес створення та підтримки веб-сайті, додатків та онлайн платформ. Це охоплює весь спектр завдань, від проектування інтерфейсу до програмування на різних мовах програмування. Веб розробники працюють над створенням сучасних та привабливих сайтів, які поєднують великий функціонал, красу та зручність у використанні. Веб-сайти, відіграють важливу роль в сучасному світі, охоплюючи широкий список сфер: електронна комерція, засоби масової інформації, освіта державні підприємства. Дозволяють організаціям та іншим користувачам ефективно та легко взаємодіяти між собою, або в середині організації. В XXI-му столітті, веб-ресурси відіграють дуже важливу роль у житті більшості людей. Вони стали частиною повсякденного життя пересічної людини. Створення веб-ресурсу – це складний процес, який включає в себе ретельне планування, вибір дизайну, розробку, тестування.

Веб-розробка є однією з найбільших сфер у сучасному світі. Практично кожна компанія, організація чи підприємство мають якісний веб-сайт, щоб бути

конкурентоспроможними. Володіння різними мовами програмування дає розробникам великі можливості для створення ефективних веб-додатків. Кожна мова програмування має свої сильні та слабкі сторони, такі як простота використання, гнучкість, ефективність, або спеціалізація в певних сферах.

Мова програмування – це штучна мова, яка слугує для передачі команд машинам, комп'ютерам. Вона схожа на людську, але з визначеними правилами та синтаксисом, які комп'ютер може зрозуміти. За допомогою мов програмування люди створюють програми, які виконують різноманітні завдання, від простих обчислень до складних веб-сайтів та штучного інтелекту.

У веб-розробці існує кілька провідних мов програмування. Серед найпопулярніших Python, JavaScript, PHP та Ruby. Кожна з цих мов має свої унікальні характеристики, такі як синтаксис, швидкодія, доступність бібліотек, спільнота розробників, підтримка фреймворків. Розглянемо кожну з цих мов програмування детальніше.

- Python – це потужна та універсальна мова програмування, яка стає популярнішою для багатьох сфер. Вона відрізняється від інших легким та зрозумілим синтаксисом, великою кількістю готових бібліотек і фреймворків, такими як Django та Flask, а також широкою сферою застосування – від створення веб-сайтів до аналізу даних та машинного навчання. Python має велику та активну спільноту розробників, що забезпечує постійну підтримку, ресурси та нововведення.
- JavaScript – мова програмування, яка використовується як на стороні клієнта, так і на стороні сервера. Надає розробникам широкі можливості для створення динамічних веб-сторінок, включаючи анімації, ефекти, обробку подій. Завдяки великій кількості бібліотек та фреймворків, таких як React, Angular та Vue.js, ця мова стала незамінною для сучасної веб-розробки.
- PHP – це одна з найстаріших та найвідоміших мов програмування для веб-розробки. Ця мова використовується для створення динамічних веб-сайтів, інтернет-магазинів, блогів, форумів та інших веб-додатків. Сьогодні PHP

продовжує розвиватись та залишається однією з найпопулярніших мов програмування для веб-розробки.

- Java – потужна мова для розробки веб-додатків за допомогою таких фреймворків, як Spring або JavaServer Faces (JSF), щоб створювати потужні масштабовані програми, які працюють на різних платформах.

Кожна з розглянутих мов програмування має свої переваги та можливості для розробки. Python пропонує розробникам легкий синтаксис і відомі фреймворки Django і Flask. JavaScript є стандартом для створення динамічних веб-сторінок.

Ці мови програмування також широко використовуються для розробки систем обліку робочого часу, необхідних для управління ресурсами та підвищення ефективності організації. Завдяки потужній бібліотеці та фреймворку розробники можуть створювати інтуїтивно зрозумілу та функціональну систему, яка автоматизує процеси управління часом, відстеження завдань та управління проектами. Вибір мови програмування такої системи залежить від конкретних потреб організації та вимог до її функціональності та інтеграції з існуючими системами. Незалежно від вибраної мови програмування, всі вони мають неабиякі можливості для створення якісних веб-додатків.

2.2 Концепція API

Також ми розглянули і інші технології без яких розробка веб-сайтів майже неможлива, сюди входить SQL, CSS, HTML та фреймворк Flask, який ми використали.

API, або прикладний програмний інтерфейс, це набір протоколів, інструментів і методів взаємодії між різними програмними компонентами. Це дозволяє розробникам отримувати доступ до функцій і даних інших програм, служб або операційних систем, не заглиблюючись у деталі їх реалізації. Сучасні API дають змогу створювати масштабовані, гнучкі та сумісні програмні рішення, що відкриває широкі можливості для інновацій та інтеграції.

API (інтерфейс прикладного програмування) це поняття з'явилося в 1960-х роках, коли комп'ютерні системи почали обмінюватися даними між собою. Однак справжній розквіт API припав на 2000-ті роки з розвитком Інтернету та веб-технологій. Такі відомі компанії, як Google, Amazon і Twitter, першими створили відкриті API для сервісів, щоб розробники могли інтегрувати ці сервіси у свої програми.

Спочатку Api був дуже примітивним і орієнтованим на просту взаємодію між програмами. Але з часом вони стали більш складними та функціональними, охоплюючи різні сфери: від платежів і аналітики до машинного навчання та хмарних технологій. Сьогодні API є невід'ємною частиною сучасної веб-архітектури, забезпечуючи гнучку та ефективну взаємодію між великими програмними компонентами.

Розробка API тісно пов'язана з розвитком мережевих технологій, таких як HTTP, REST і JSON. Цей API був більш стандартизованим, масштабованим і доступним для різних середовищ, включаючи мобільні пристрої та хмарні обчислення. Сучасні API також приділяють велику увагу питанням безпеки та авторизації, щоб захистити дані та функції від несанкціонованого доступу.

Існує декілька основних типів API, такі як:

- **REST** - це архітектурний стиль для створення веб-сервісів, заснований на використанні протоколу HTTP для обміну даними між клієнтом і сервером. REST API використовує стандартні методи HTTP (GET, POST, PUT, DELETE) для взаємодії з ресурсами, ідентифікованими за допомогою унікальних URL-адрес. Це дозволяє створити гнучкий, масштабований і простий для розуміння Api.

- **Soap** - це протокол, який використовує XML для кодування запитів та відповідей. SOAP Api використовує стандартні веб-технології, такі як HTTP та SMTP, для пересилання службових повідомлень. SOAP API зазвичай забезпечують більш суворий контроль даних та їх безпеку, але їх може бути складніше розробляти та інтегрувати.

- **GraphQL**-це мова запитів API та середовище виконання, призначені для взаємодії клієнт-сервер. На відміну від традиційних API REST, GraphQL зменшує кількість переданих даних, оскільки клієнти можуть запитувати лише потрібну інформацію. Це робить API GraphQL більш ефективним та гнучким, особливо для складних додатків з великою кількістю релевантних даних.

Розвиток технологій і зростання потреб бізнесу привели до швидкої еволюції API в сучасному світі. Серед основних тенденцій можна виділити наступні: розробка окремих взаємопов'язаних служб, які взаємодіють через API, стає все більш поширеною, оскільки дозволяє гнучко масштабувати програми, компанії починають розробляти продукти і сервіси на основі API-інтерфейсу, орієнтуючись на потреби користувачів, API забезпечують гнучку, масштабовану та надійну інтеграцію різних сервісів, і тому такі архітектури стають все більш популярними. В цілому, сучасна розробка API характеризується підвищеною гнучкістю, масштабованістю і автоматизацією, що дозволяє компаніям швидко адаптуватися до мінливих вимог ринку.

Розробка та документування API-інтерфейсів є важливим етапом у створенні програмного забезпечення, що взаємодіє із зовнішніми системами. Цей процес зазвичай починається з ретельного проектування API, визначення кінцевих точок, параметрів та структури відповідей. За цим слідує реалізація серверної частини, яка обробляє запити користувачів і видає відповідні результати.

Важливим кроком є всебічне тестування API для забезпечення стабільної та коректної поведінки. Після завершення розробки потрібно буде створити детальну документацію, в якій будуть описані всі функції API, їх структура, порядок їх затвердження, приклади використання та багато іншого. Такі документи полегшують інтеграцію з API для сторонніх розробників і забезпечують довгострокову підтримку програмних продуктів.

Інтеграція API з веб-фреймворками, такими як Flask, є важливим аспектом сучасної веб-розробки. Flask – це легкий, гнучкий і зручний у використанні Python-фреймворк, що ідеально підходить для розробки api-додатків. Він

забезпечує простий та ефективний спосіб створення маршрутів, обробки запитів та відповідей, а також інтеграції з базами даних та іншими службами.

Взаємодія між Flask та API включає кілька важливих етапів. По-перше, потрібно визначити маршрут для доступу до кінцевої точки API за допомогою функції Flask, наприклад @app.Route. Потім потрібно реалізувати логіку, яка обробляє запит і генерує відповідь як JSON або інший формат. Flask надає можливість легко інтегрувати зовнішні API через бібліотеки, такі як requests, і обробляти аутентифікацію і авторизацію користувачів.

Взаємодія API з Flask дає розробникам гнучкий контроль над процесом створення надійних і масштабованих веб-додатків. Поєднуючи потужність Flask з простотою роботи з API, ви можете швидко створювати сучасні веб-рішення, які можуть бути легко інтегровані з іншими сервісами.

З розвитком технологій та зростаючими потребами користувачів майбутнє API виглядає особливо перспективним. Однією з основних тенденцій є подальше поширення архітектур мікросервісів, посилення інтеграції API з різними пристроями (IoT), а також забезпечення безпеки та доступу до API за допомогою машинного навчання та штучного інтелекту для автоматизації процесів управління API та аналізу даних. Очікується, що Api стане ще більш орієнтованим на користувача завдяки більш доступній документації та кращим інструментам тестування та моніторингу. Також можна прогнозувати збільшення числа відкритих API, що сприятиме інноваціям і розвитку екосистеми, що оточує різні продукти і сервіси.

2.3 Фреймворк Flask

Flask — це простий, але потужний веб-фреймворк Python, який надає розробникам гнучкі інструменти для створення веб-додатків будь-якого розміру. На відміну від деяких важких і складних фреймворків, Flask має мінімальну структуру, яка дозволяє швидко розробляти та розгортати веб-додатки.

Фреймворк був створений Armin Ronacher в 2010 році і орієнтований на швидку розробку прототипів і невеликих веб-додатків. Flask дає розробникам гнучкість у виборі бібліотек та інструментів, необхідних для реалізації конкретних вимог проекту. Це робить його ідеальним вибором для створення API, мікросервісів, блогів або невеликих веб-порталів.

Flask надає лише основні інструменти для розробки веб-додатків, включаючи маршрутизацію URL, обробку запитів та шаблони для генерації HTML. Завдяки простому і зрозумілому API, Flask легко освоїти навіть початківцям. Розробники можуть швидко створити перший веб-додаток, що робить його ідеальним для навчання та прототипування. Однією з основних переваг Flask є його здатність забезпечувати швидке прототипування. Завдяки своїй простоті та швидкості Flask часто використовується для створення прототипів та Mvp (мінімально життєздатних продуктів). Це дозволяє розробникам швидко тестувати свої ідеї та отримувати зворотній зв'язок. Також чудовий інструмент для навчання веб-розробці. Його простота дозволяє новачкам зосередитися на основних поняттях веб-розробки, не обтяжуючи їх складними інструментами і конфігураціями. Це робить Flask популярним вибором для навчальних програм та курсів веб-розробки. Ще одна важлива особливість Flask — гнучкість у виборі інструментів. Flask не містить жодної конкретної бібліотеки чи інструментів, тому розробники можуть вільно вибирати найкращу технологію для конкретного проекту. Це особливо корисно для проектів, які потребують використання певних рішень або інтеграцій.

```
136
137 @app.route('/')
138 def home():
139     return 'Hello, World!'
140
141 if __name__ == '__main__':
142     app.run(debug=True)
143
```

Рисунок 2.1 – Приклад синтаксису Flask

У цьому прикладі зображеному на рисунку 2.1 створюється веб-програма, яка обробляє HTTP-запити до кореневої URL-адреси (/) і повертає текстове повідомлення «Hello, World!». Простий приклад демонструє, як легко створити цю веб-програму за допомогою Flask, використовуючи мінімальний код.

Фреймворк Flask підтримує всі основні принципи розробки веб-додатків, такі як архітектура Model-View-Controller (MVC), яка допомагає структурувати ваш код. Flask не нав'язує конкретну архітектуру, але розробники можуть легко слідувати вказівкам і найкращим практикам. Це дозволяє вам бути впевненим, що ваш код чистий і його можна підтримувати.

Таким чином, Flask – це потужний та гнучкий інструмент веб-розробки, який підходить для різних типів проектів, від простих веб-сайтів до складних веб-додатків та мікросервісів. Його простота та масштабованість роблять його привабливим варіантом для розробників, які прагнуть створити ефективні та надійні рішення. Завдяки активній спільноті та широкому спектру доступних ресурсів, Flask продовжує залишатися однією з найпопулярніших платформ веб-розробки Python.

2.4 Стек технологій для розробки

Розробка веб-додатків використовує широкий спектр технологій, кожна з яких відіграє важливу роль у створенні функціональних та ефективних продуктів. У цьому розділі описано стек технологій, який включає html, CSS, JavaScript, Python, Flask, SQL і Google Sheets API.

HTML, мова гіпертекстової розмітки, служить основною мовою для створення веб-сторінок, дозволяючи змінювати структуру документа та вміст, що відображається в браузері, і служить основною технологією Всесвітньої павутини. HTML є основною мовою для створення всіх веб-сторінок. Він пропонує базову структуру для створення та публікації вмісту в Інтернеті. Без HTML веб-розробники не змогли б створити сайти, програми та інші

інтерактивні ресурси, які ми використовуємо сьогодні. HTML-документ складається з елементів, укладених у теги. Теги зазвичай бувають парами з початковим і закриваючим тегами. Однак деякі теги самозакриваються. Крім того, теги можуть мати атрибути, які надають додаткову інформацію про елемент. Ці атрибути розміщуються у початковому тегу та відповідають формату name="value".

Основні теги в HTML поділяються на:

- Структурні теги. HTML-документ складається з основних структурних тегів, які утворюють частини веб-сайту, наприклад заголовок, основний вміст, навігацію та нижній колонтитул. Це теги <html>, <head>, <body>, <header>, <nav>, <main>, <section>, <article>, <aside>, <footer>, .
- Мультимедійні теги. Теги <img>, <video> і <audio> використовуються для додавання зображення, відео та аудіо на сторінку. Ці теги можуть додавати мультимедійний контент і досягати його відображення на сторінці.
- Теги форматування. Для форматування тексту на веб-сторінці використовується широкий спектр тегів, наприклад <h1> - <h6> для заголовків, <p> для абзаців, <b>, <i>, <u> для виділення тексту.

Правильне використання семантичних тегів гарантує, що код HTML є структурованим і значущим, точно передає інформацію вмісту. Це вкрай важливо для розробки високоякісних, доступних веб-сайтів, які відповідають сучасним стандартам веб-розробки.

Кожен документ HTML має певну структуру з основними компонентами, які допомагають браузерам правильно відображати та інтерпретувати веб-сторінки. <!DOCTYPE> з'являється на початку документа, інформуючи браузер про тип документа та правила інтерпретації HTML-коду. Весь вміст веб-сторінки укладено в теги <html>, які складаються з двох основних розділів: <head> і <body>. Розділ <head> містить метадані та налаштування для сторінки, наприклад заголовок сторінки, стилі CSS, сценарії JavaScript тощо. Ці дані не відображаються на сторінці. Основний розділ <body> містить увесь видимий

вміст веб-сторінок, наприклад текст, зображення, посилання, форми тощо. Дотримання стандартів структурування документів HTML забезпечує семантичну точність, доступність і сумісність сторінок.

```html
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>HTML </title>
</head>
<body>
    <h1>Привіт, світ!</h1>
</body>
</html>
```

Рисунок 2.2 – Приклад синтаксису HTML

На рисунку 2.2 зображено найпростішу веб-сторінку. HTML-форми є важливим інструментом веб-розробки для збору інформації та взаємодії з користувачами. Ці форми містять такі елементи, як текстові поля, розкривні списки, кнопки та прапорці, що дозволяє користувачам вводити, вибирати та надсилати дані на сервер. Ці дані можна використовувати для різних цілей, таких як реєстрація, вхід, замовлення продуктів, відгуки та опитування.

Щоб створити форму, потрібно зрозуміти теги HTML, такі як <form>, <input>, <textarea>, <select>, <button> тощо. Кожен тег має власні атрибути, які сприяють його функціональності та зовнішньому вигляду. Розробники також використовують CSS для розробки форм і JavaScript для інтерактивності на стороні клієнта, перевірки та обробки даних. Правильне використання форм HTML має вирішальне значення для створення зручного та ефективного інтерфейсу користувача.

Таким чином, HTML є основою веб-розробки та забезпечує структуру, семантику та основні функції веб-сторінок. HTML разом із CSS і JavaScript відіграє важливу роль у створенні сучасних, функціональних і привабливих веб-додатків, які відповідають потребам користувачів.

CSS (каскадні таблиці стилів) — це стилістична мова дизайну, яка використовується для опису зовнішнього вигляду веб-сторінок, написаних мовами розмітки HTML або XML. Відокремлюючи вміст веб-сторінки від її дизайну, веб-дизайнери зберігають повний контроль над тим, як виглядає вміст. CSS працює, створюючи правила, які застосовуються до елементів сторінки. Ці правила включають такі властивості, як колір, шрифт, розмір і положення.

CSS (каскадні таблиці стилів) має власний синтаксис, який дозволяє стилізувати елементи HTML і створювати красиві веб-сторінки. Основна структура CSS складається з селекторів і оголошень. Селектор визначає стиль елемента HTML, а оголошення містить фактичний стиль, застосований до цього елемента. Синтаксис CSS складається з таких компонентів:

1. Селектор: це визначає стиль елемента HTML і може бути тегом, класом, ідентифікатором або будь-яким іншим типом селектора.
2. Властивість: надає певні характеристики, які ви хочете змінити за допомогою елементів стилю, таких як колір, розмір шрифту та поля.
3. Значення: це визначає конкретне значення, яке застосовується до певної властивості.

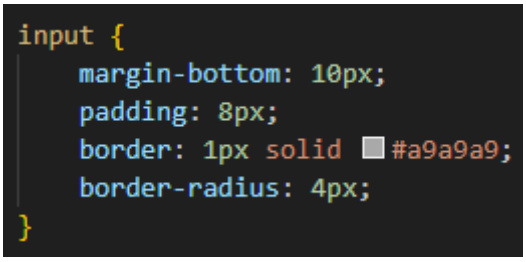

Рисунок 2.3 – Приклад синтаксису CSS

Наприклад, скажімо, колір рамки "сірий" зображену на рисунку 2.3 або розмір шрифту "16px". Оголошення складається з властивостей і значення, розділених двокрапкою (наприклад, «рамка: сіра;»). Він містить одну або більше декларацій, згрупованих за допомогою фігурних дужок {} навколо селектора блоку стилів. Правильне використання синтаксису CSS є ключовим для створення добре структурованої та стилізованої веб-сторінки. Вивчення

синтаксису допоможе вам краще зрозуміти CSS і ефективно використовувати його під час розробки веб-сайту.

Існує декілька типів підключень CSS до HTML. Внутрішнє зв’язування CSS забезпечує розміщення стилю окремо в документі HTML у спеціальному тегу <style>. Цей метод забезпечує швидкий доступ до стилів редагування, але може бути громіздким через велику кількість коду. Крім того, внутрішнє з'єднання зменшує гнучкість стилю та його повторне використання. Зовнішнє підключення для створення CSS-коду передбачає розміщення CSS-коду в окремому файлі з розширенням .css. Цей файл потім зв’язується з HTML-документом за допомогою тегу <link>. Зовнішнє посилання пропонує кращу організацію та модульність коду, полегшуючи підтримку та оновлення стилів на сайті.

Каскадність і специфічність є важливими поняттями в CSS, які визначають, який стиль буде застосовано до певного елемента на веб-сторінці. Каскадування — це правило, яке використовується веб-переглядачем, щоб визначити, який стиль має перевагу, коли те саме правило CSS застосовується до елемента кількома способами. Специфічність — це концепція визначення того, які селектори CSS мають найвищий пріоритет, коли до одного елемента застосовуються кілька правил CSS.

Каскадні правила базуються на таких факторах, як розміщення стилю (внутрішній, внутрішній або зовнішній), важливість (чи позначено стиль як важливий чи ні) і порядок, у якому вони з’являються в коді. Специфічність обчислюється за допомогою числового значення, що складається з трьох частин: ідентифікатора селектора, класу та тегу. Чим вище значення цього числа, тим вищий пріоритет селектора.

Чітке розуміння концепції каскадування та специфічності дозволяє розробникам передбачити, які стилі будуть застосовані до елементів, уникати конфліктів між стилями та гарантувати, що веб-сторінки виглядатимуть так, як вони задумали.

JavaScript — це ключова мова програмування, яка використовується для створення інтерактивних і динамічних веб-сайтів. Ця потужна мова практично інтегрована в кожну сторінку сучасного Інтернету, контролюючи все: від анімації та інтерфейсів до серверної логіки та баз даних. JavaScript надає веб-розробникам широкі можливості для створення привабливих, інноваційних і функціональних веб-додатків.

Одним із ключових принципів є об'єктно-орієнтоване програмування. JavaScript дозволяє розробникам створювати об’єкти, що містять властивості та методи, забезпечуючи гнучкість у розробці складних систем. Іншою важливою концепцією є функціональне програмування, яке зосереджується на використанні чистих функцій, які не змінюються ззовні, що призводить до більш стабільного та масштабованого коду.

Не менш важливою концепцією є асинхронне програмування. У JavaScript є вбудовані механізми, такі як зворотні виклики, обіцянки та async/await, які дозволяють програмам виконувати операції одночасно, не блокуючи кілька основних потоків виконання. Це вкрай важливо для створення інтерактивних високопродуктивних веб-додатків.

Крім того, JavaScript підтримує концепцію створення прототипів, дозволяючи розробникам створювати складні ієрархії об’єктів і повторно використовувати код. Розуміння цих та інших фундаментальних концепцій є невід’ємною частиною успішного використання JavaScript у веб-розробці.

База даних є ключовим компонентом сучасних інформаційних систем, оскільки вона забезпечує ефективне зберігання, керування та доступ до великих обсягів даних. SQL (мова структурованих запитів) — це стандартна мова для взаємодії з реляційними базами даних, що дозволяє виконувати широкий спектр операцій із даними, включаючи створення, оновлення, видалення та пошук.

Однією з основних функцій бази даних є зберігання даних. Бази даних дозволяють структуровано зберігати великий обсяг інформації. Це особливо важливо для організацій, які мають справу з великими обсягами даних, таких як

корпорації, медичні заклади, фінансові установи та державні організації. Зберігання даних у базах даних забезпечує їх цілісність, безпеку та доступність.

Бази даних ефективно керують даними, обробляючи додавання, оновлення, видалення та пошук. Крім того, вони пропонують контроль доступу для захисту даних і забезпечення конфіденційності. Бази даних дозволяють веб-додаткам обробляти велику кількість користувачів і запитів за допомогою індексів, кешування запитів і розділення бази даних для підвищення продуктивності. Ці методи забезпечують швидкий доступ до даних і зменшують затримку.

SQL є основною мовою, яка використовується для взаємодії з реляційними базами даних у веб-розробці. Веб-додатки використовують SQL для запитів до баз даних для завантаження, вставки, оновлення та видалення даних. Наприклад, запити SQL використовуються для автентифікації користувача, пошуку продуктів, замовлення та аналізу даних. Однією з найпоширеніших загроз безпеці веб-додатків є впровадження SQL. Ці атаки відбуваються, коли зловмисники вводять шкідливий код SQL через введення користувача, щоб отримати несанкціонований доступ до даних або маніпулювати ними. Щоб захистити веб-додатки від SQL-ін’єкцій, розробники повинні вживати належних заходів безпеки. Шифрування є важливим методом захисту даних користувача. Веб-додатки повинні використовувати шифрування для захисту конфіденційної інформації, такої як паролі, фінансові відомості та особисті дані. Запобігти несанкціонованому доступу до конфіденційної інформації допоможе використання сучасних алгоритмів шифрування та забезпечення безпечного зберігання ключів шифрування.

Існує декілька основних типів нормалізації бази даних, сюди входить:

1. Перша нормальна форма. Перша нормальна форма встановлює фундаментальний спосіб організації даних, запобігаючи появі повторюваних груп і полегшуючи доступ до окремих значень.

1.1. Кожен стовпець таблиці повинен містити тільки неділимі значення.

1.2. Кожен стовпець повинен містити тільки один тип даних.

1.3. Кожен рядок таблиці повинен бути унікальним.

2. Друга нормальна форма. Друга нормальна форма прибирає часткові залежності, які з’являються, коли неключові атрибути залежать тільки від частини складеного первинного ключа.

2.1. Кожен неключовий стовпець повинен повністю залежати від первинного ключа.

2.2. Повинна відповідати вимогам першої нормальної форми.

3. Третя нормальна форма. Третя нормальна форма гарантує, що неключові атрибути не залежать один від одного через первинний ключ.

3.1. Кожен неключовий стовпець повинен залежати тільки від первинного ключа.

3.2 Повинна відповідати вимогам другої нормальної форми.

4 .Бойс-Кодд нормальна форма. Вирішує деякі особливі випадки аномалії 3NF, особливо у випадку, коли таблиці мають більше одного кандидата в ключі.

4.1 Кожен визначник повинен бути кандидатом в ключ.

5.Четверта нормальна форма. Усуває багатозначні відносини, що відрізняються від того, коли один атрибут залежить від іншого через множинні значення.

5.1 Жоден неключовий стовпець не повинен залежати від багатьохзначних залежностей.

6. П'ята нормальна форма. Розбиває таблиці для усунення складних залежностей, які можуть викликати аномалії приєднання, коли дані розділяються на більш дрібні частини.

6.1 Жоден неперехідний залежний неключовий атрибут не повинен бути залежним від кількох значень первинного ключа.

Нормалізація бази даних є важливим процесом, який забезпечує цілісність даних, зменшує неефективність і підвищує ефективність керування даними. Кожен етап нормалізації усуває певні аномалії та залежності, допомагаючи створити оптимальну структуру бази даних для різних програм.

РОЗДІЛ 3
РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ РОБОЧОГО ЧАСУ ПРАЦІВНИКІВ ВІДДІЛУ КІТ

Враховуючи сучасні умови праці, облік робочого часу співробітників є найважливішим аспектом управління персоналом. Однак використання паперових журналів і ручних методів обліку призводить до багатьох проблем. Ці методи неефективні, оскільки потребують значного часу та зусиль для запису та обробки даних. Людські помилки можуть призвести до неточностей в обліку робочого часу, що призведе до неправильного розрахунку заробітної плати та інших важливих показників. Крім того, цей ручний процес обліку та аналізу даних займає багато часу та негативно впливає на ефективність відділу кадрів. Відсутність оперативного доступу до актуальної інформації ускладнює прийняття управлінських рішень та контроль за дотриманням трудової дисципліни.

Мета розробки інформаційної системи обліку робочого часу працівників відділу передбачає автоматизацію процесу зберігання обліку робочого часу. Це спрямовано на підвищення точності даних та ефективності управління персоналом. Система повинна забезпечувати легке і швидке ведення, зберігання та обробку даних про робочий час співробітників, зводячи до мінімуму ризик помилок. Ключовим пріоритетом є забезпечення легкого доступу до інформації для керівництва, що сприяє швидкому доступу до необхідних даних.

3.1 Архітектура інформаційної системи

Архітектура інформаційної системи відноситься до структурної композиції компонентів програмного та апаратного забезпечення, призначених для досягнення конкретних цілей, а також до їх взаємодії та взаємозв’язків. У сучасному швидкоплинному та конкурентному бізнес-середовищі архітектура

інформаційних систем відіграє вирішальну роль у забезпеченні успіху, забезпечуючи швидкість та ефективність.

Термін «випадки використання» відноситься до послідовності дій, які виконує система у відповідь на вхідні дані користувача або програмної системи. Ці варіанти використання представляють функціональні можливості системи з орієнтованої на користувача перспективи та допомагають визначити пріоритети функціональних можливостей на основі бажаних результатів. Цей процес необхідний для визначення функціональних вимог системи та управління її розвитком. Такі базові дії, як аналіз, проектування та тестування, базуються на цих варіантах використання. На етапах аналізу та проектування ми перевіряємо, як бажані користувачем результати впливають на архітектуру системи та як компоненти системи повинні функціонувати, щоб забезпечити бажану функціональність.

Діаграма варіантів використання зазвичай включає суб’єкта (систему), який виконує дію, і дію варіантів використання, яка описує, чого суб’єкт очікує від системи. Щоб зрозуміти, як користувачі взаємодіють із системою, нам потрібно розглянути та відповісти на запитання про користувачів, які взаємодіють із системою, тих, хто надсилає та підтримує інформацію в системі, і тих, хто є зовнішніми щодо системи.

Загалом, діаграма випадків на рисунку 3.1 сприяє кращому розумінню потреб користувачів, визначенню функціональних вимог до системи та її архітектурі, що робить їх важливим інструментом при проектуванні інформаційних систем.

Блок-схема є важливою складовою при розробці інформаційної системи обліку робочого часу співробітників IT-департаменту. Він представляє структуру та взаємодію всіх компонентів системи, дозволяючи візуалізувати архітектурну систему та демонструвати, як різні компоненти взаємодіють один з одним, включаючи клієнтську сторону (інтерфейс), серверну сторону (бекенд) і базу даних. Ця візуалізація допомагає розробникам і зацікавленим сторонам краще зрозуміти загальну архітектуру системи.

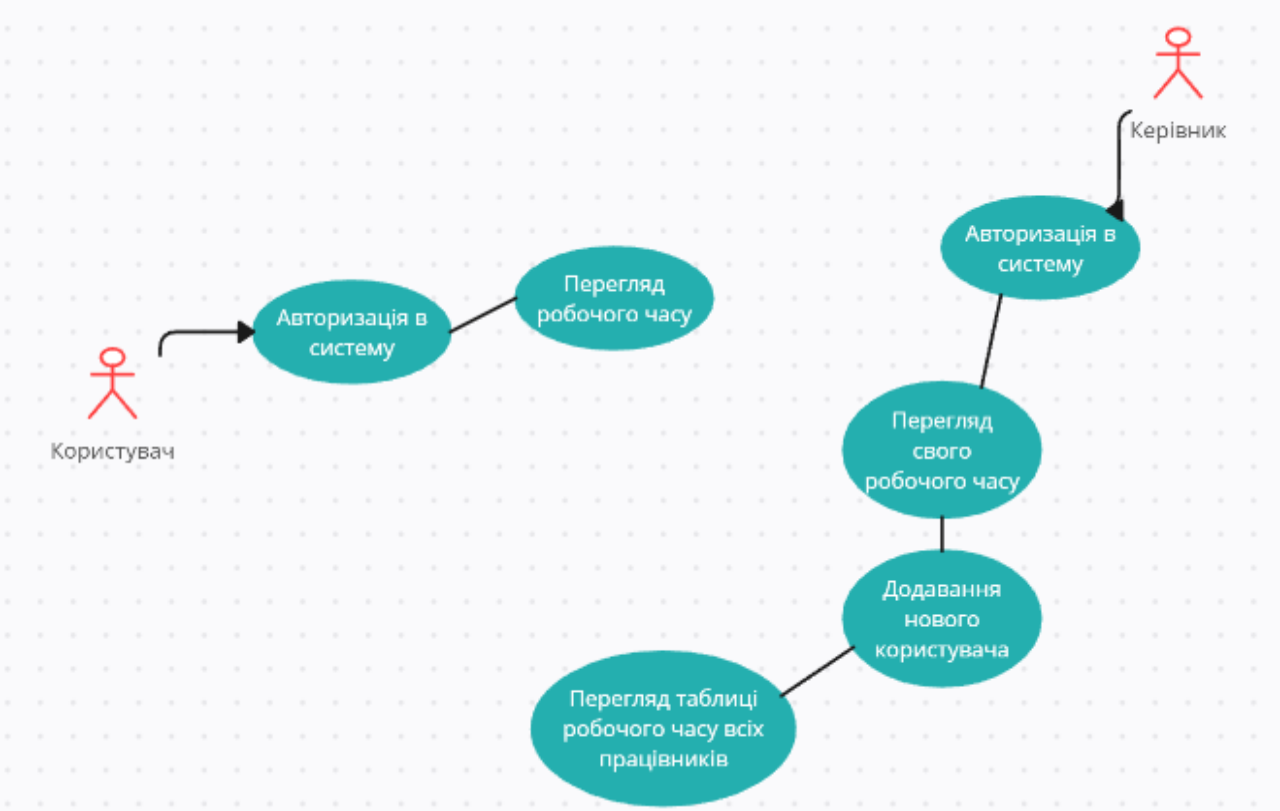

Рисунок 3.1 – Use Case діаграма

Блок-схема спрощує процес розробки, дозволяючи розробникам краще планувати та координувати свою діяльність і забезпечуючи чітке визначення того, які частини системи необхідно впровадити та як ці частини будуть взаємодіяти. Це полегшує спілкування між учасниками проекту, слугуючи спільною мовою для розробників, менеджерів і клієнтів, знижуючи ризик непорозумінь.

Блок-схема допомагає виявляти та усувати дефекти на ранніх стадіях розробки, полегшуючи виявлення таких проблем, як неефективна взаємодія між компонентами або виклики критичних функцій. Він також служить елементом системи технічної документації, допомагаючи новим членам команди швидко ознайомитися з архітектурою та принципами системи.

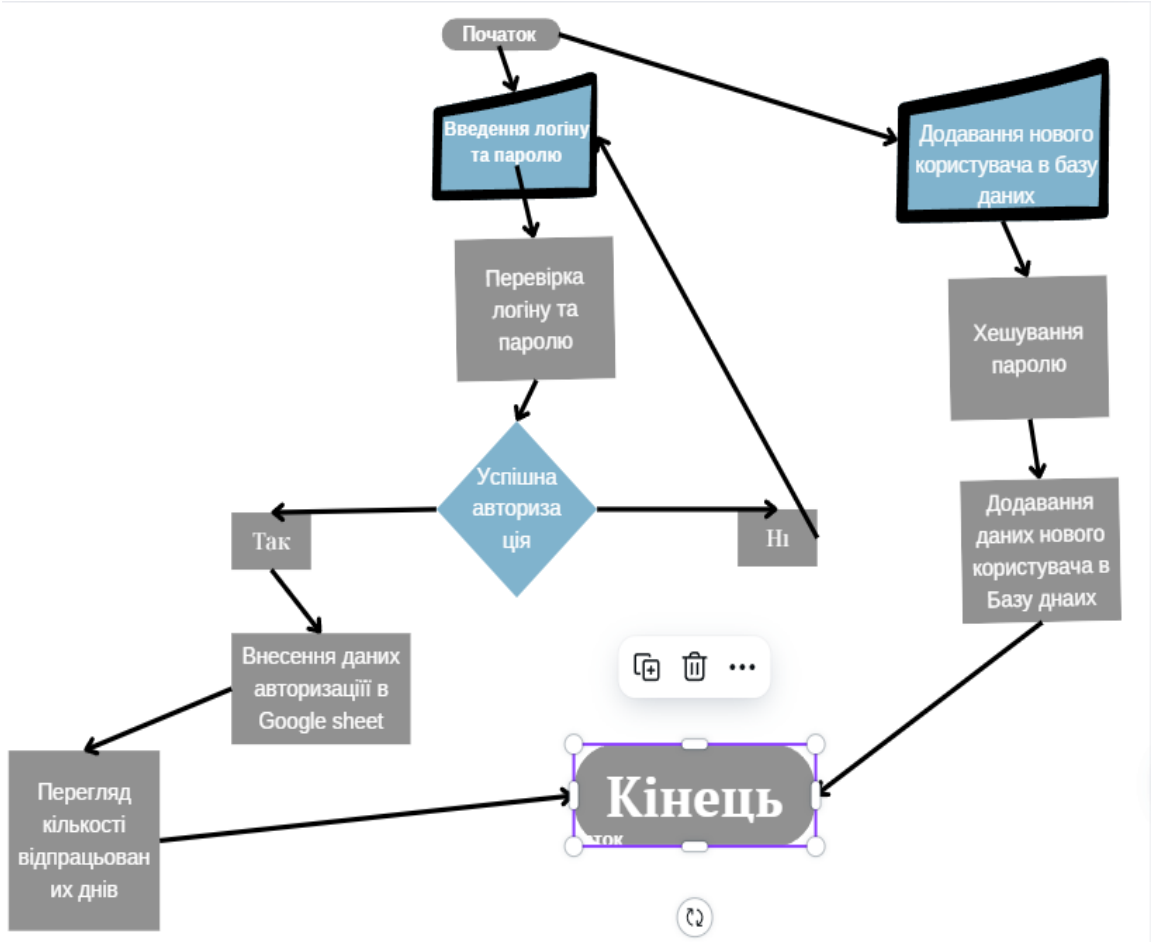

Рисунок 3.2 – Блок-схема системи

Таким чином, блок-схема зображена на рисунку 3.2 є невід'ємною частиною розробки інформаційної системи обліку робочого часу співробітників відділу КІТ, що забезпечує ефективність і зручність на етапах проектування, розробки, тестування та підтримки системи.

3.2 Клієнтська частина інформаційної системи

Склад та функціонал інформаційної системи обліку робочого часу, включаючи HTML сторінки, CSS та JavaScript код для веб-інтерфейсу, серверний код на базі Flask, базу даних для зберігання інформації про користувачів та їх робочий час, а також інтеграцію з API Google Sheets для додаткової зручності та забезпечення надійності даних.

Користувальницький компонент інформаційної системи має вирішальне значення для забезпечення зручної та ефективної взаємодії користувача з системою. Він служить інтерфейсом, через який користувачі можуть отримати доступ до функцій системи, взаємодіяти з даними та виконувати різні завдання.

Компонент користувача може приймати різні форми, включаючи веб-інтерфейси, настільні програми, мобільні програми та голосові помічники. Він послідовно враховує потреби та переваги різних груп користувачів, щоб забезпечити зручність і доступність інтерфейсу.

Спеціальний компонент може містити такі елементи, як навігаційні меню, форми введення даних, списки, таблиці, графіки та інші візуальні елементи для представлення інформації. Він також може охоплювати функції пошуку та фільтрації даних, налаштування користувацьких параметрів, надсилання повідомлень тощо.

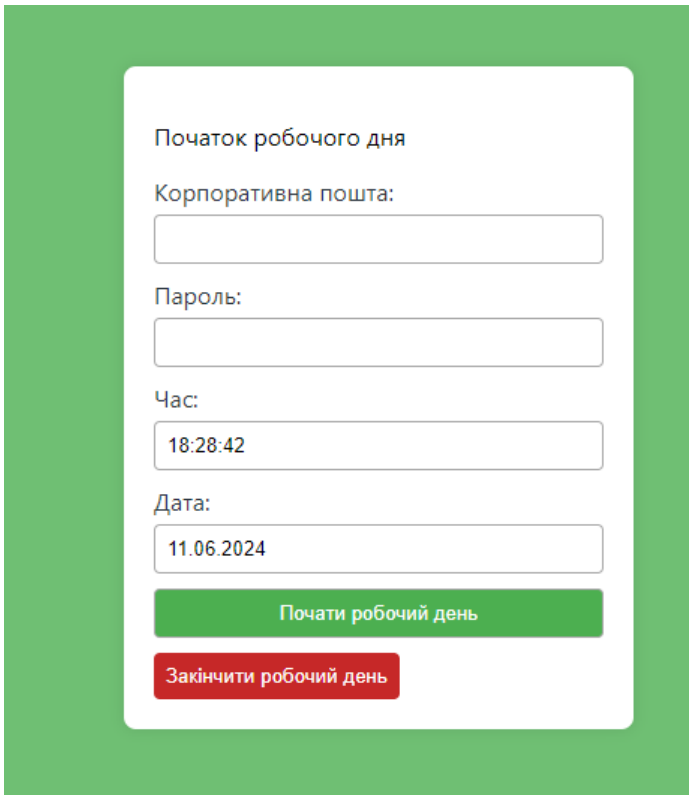

Рисунок 3.3 – Сторінка авторизації

На рисунку 3.3 зображено форму, яка використовується для входу в інформаційну систему. Сторінка містить два поля для вводу даних, перше поле для вводу корпоративної пошти, друге поле для вводу паролю користувача. Також є ще 2 поля це дата та час авторизації користувача. Дві кнопки які відповідають за вхід в систему та вихід зі сторінки.

Загалом користувацький компонент інформаційної системи відіграє життєво важливу роль у задоволенні потреб користувачів і досягненні цілей системи. Це дозволяє користувачам ефективно використовувати системні ресурси, отримувати доступ до необхідної інформації та виконувати різноманітні завдання.Інформаційна система на даному етапі містить 3 HTML сторінки. Які стилізовані за допомогою CSS.

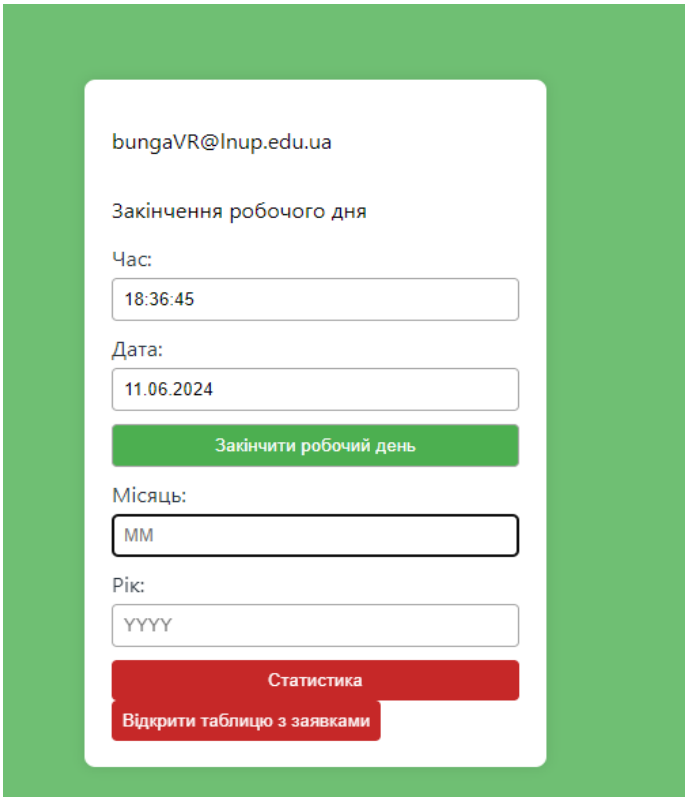

Рисунок 3.4 – Основна сторінка

Сторінка, яка зображена на рисунку 3.4 містить кнопку «Статистика» та «Закінчити робочий день» для виходу зі сторінки. Ця сторінка створена для виходу з системи та перегляду відпрацьованих днів за певний місяць. Тобто

користувач після авторизації може ввести місяць та рік та отримати кількість відпрацьованих днів

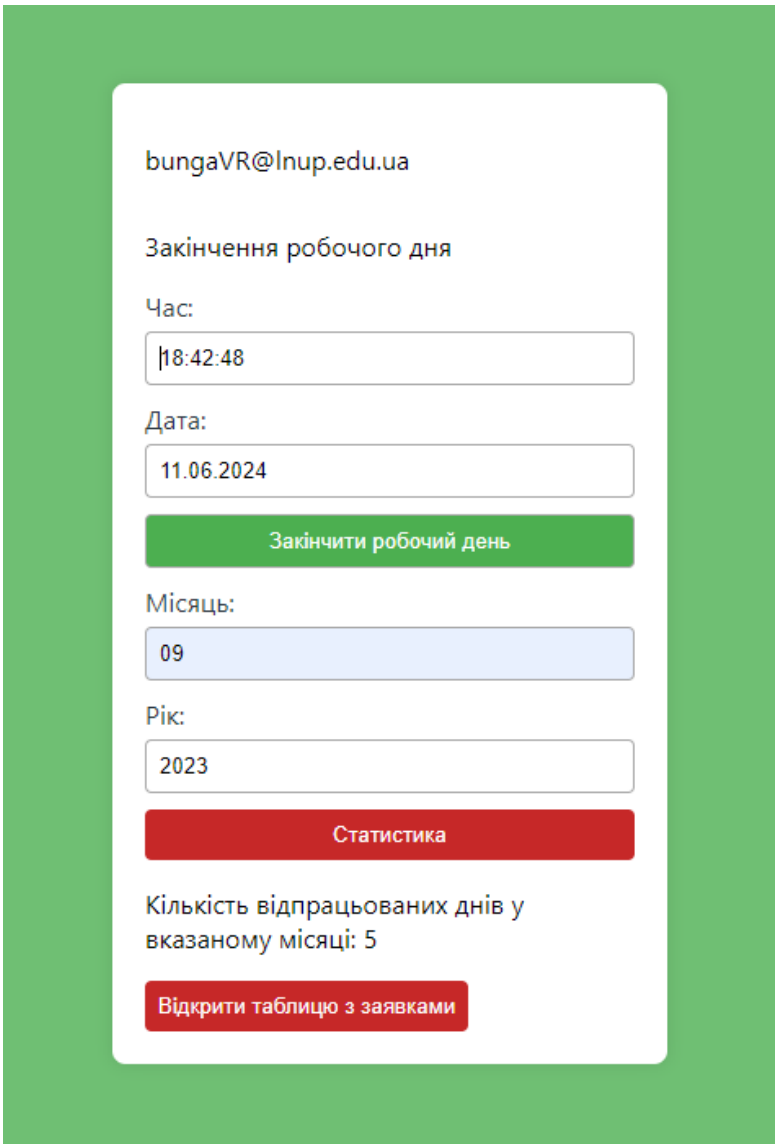

Рисунок 3.5 – Статистика відпрацьованих днів

На на рисунку 3.5, користувач може переглянути сторінку з кількістю відпрацьованих днів за певний місяць року. Також сторінка містить кнопку «Відкрити таблицю з заявками», після натискання на неї ми перейдемо до таблиці, яка містить дані про заявки відділу КІТ.

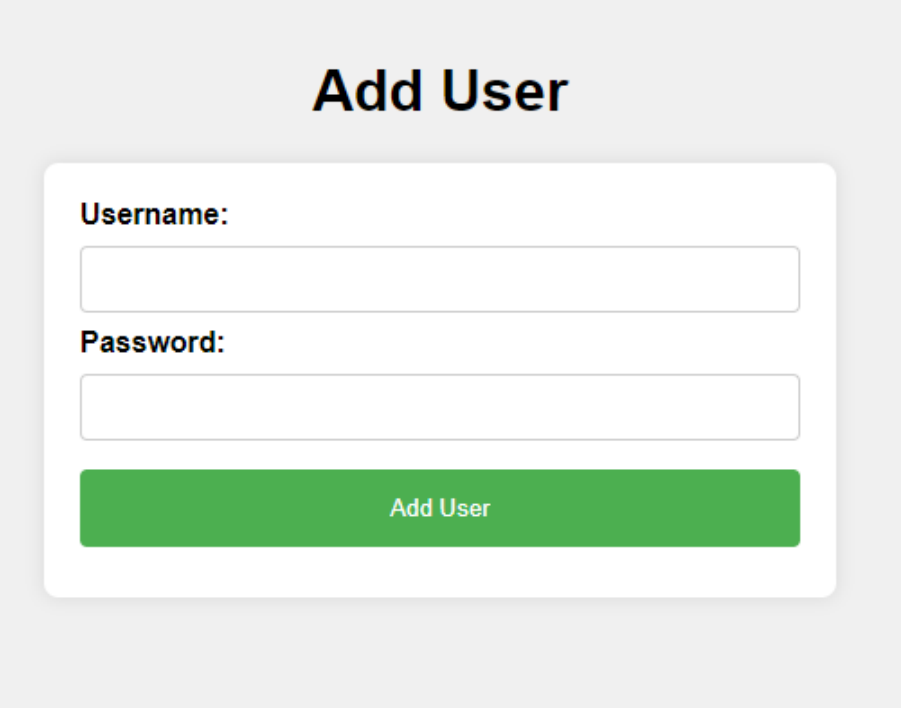

Рисунок 3.6 – Панель для створення нових користувачів

Сторінка для створення нових користувачів до попередньо створеної бази даних. На рисунку 3.6 зображено два поля «Username» для користувача, «Password» для створення паролю користувача.

3.3 Серверна частина інформаційної системи

Серверна частина інформаційної системи відіграє важливу роль у забезпеченні ефективного функціонування та обробки даних. Це центральний елемент, який забезпечує зв'язок та обмін інформацією між усіма компонентами системи та її користувачами.

Ця частина системи виконує обробку запитів, забезпечує доступ до даних, що зберігаються в базах даних і ресурсах, деякі для виконання різних операцій. Його основне завдання — забезпечення безпеки даних, масштабованості та надійності всієї системи.

Після вводу даних на сторінці, яка зображена на рисунку 3.3 відбувається перевірка відповідності введених даних, якщо дані правильні, то системи

відкриває нам наступну сторінку зображену на рисунку 3.4, якщо дані неправильні, система повертає помилку.

```python
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        email = request.form['username']
        password = request.form['password']
        current_time = request.form['time']
        current_date = request.form['date']

        conn = sqlite3.connect('D:\\learning\\dyplom\\instance\\user.db')
        cursor = conn.cursor()
        cursor.execute('SELECT * FROM Users WHERE username = ?', (email,))
        user = cursor.fetchone()
        conn.close()

        if user and check_password_hash(user[2], password):
            session['user_id'] = user[0]
            flash('Успішний вхід!', 'success')
            add_data_to_google_sheet(email, current_time, current_date)
            return redirect(url_for('end_work_day'))
        else:
            flash('Неправильний логін або пароль.', 'danger')
            print("Не вдалося увійти")

    return render_template('index.html')
```

Рисунок 3.7 – Фрагмент коду для авторизації

Наведений фрагмент коду, зображений на рисунку 3.7 є частиною веб-програми Flask, яка керує автентифікацією користувачів і записує робочі години в Google Sheet.

Декоратор @app.route('/', methods=['GET', 'POST']) встановлює маршрут для домашньої сторінки та пов'язує його з функцією index(). У функції index() перевіряється, чи метод HTTP є POST. Якщо це так, функція отримує ім’я користувача, пароль, поточний час і поточну дату з даних форми.

Якщо користувача знайдено в базі даних і пароль збігається з хешованим паролем, що зберігається в базі даних (перевіряється за допомогою check_password_hash()), ідентифікатор користувача зберігається в сеансі Flask. Відображається миттєве повідомлення про успіх і викликається функція add_data_to_google_sheet(), щоб зареєструвати робочі години в таблиці Google. Нарешті, користувач перенаправляється на сторінку end_work_day().

Якщо користувача не знайдено або пароль не збігається, відображається миттєве повідомлення про небезпеку, і користувач повертається на головну сторінку.

```python
def add_data_to_google_sheet(email, current_time, current_date):
    username , domain = email.split("@")
    match email:
        case "bungaVR@lnup.edu.ua":
            pib = "Бунга В.Р."
        case "vasilishinTV@lnup.edu.ua":
            pib ="Василишин Т.В."
        case "shuvarbi@lnup.edu.ua":
            pib = "Шувар Б.І."
        case "homyakNY@lnup.edu.ua":
            pib = "Хом'як Н.Ю."
        case "stankoVY@lnup.edu.ua":
            pib = "Станько В.Ю."


    try:
        worksheet = client.open("Workdays").worksheet(username)
        next_row = len(worksheet.col_values(1)) + 1
        data = [pib, email, current_time, current_date]
        worksheet.append_rows([data], value_input_option='USER_ENTERED', insert_data_option='INSERT_ROWS', table_range=f'A{next_row}:C{next_row}')
    except gspread.exceptions.WorksheetNotFound as e:
        print(f"Аркуш не знайдено. Помилка: {e}")
    except exceptions.GoogleAuthError as e:
        print(f"Помилка аутентифікації Google: {e}")
    except Exception as e:
        print(f"Інша помилка: {e}")
```

Рисунок 3.7 – Фрагмент коду запису даних Google sheet

Функція спочатку, яка зображена на рисунку 3.7 розділяє електронну пошту на ім’я користувача та домен за допомогою символу «@» та намагається відкрити таблицю Google під назвою "Workdays" та отримати доступ до робочої таблиці, яка відповідає імені користувача. Дані, які потрібно додати до таблиці Google, зберігаються в списку.

Потім функція додає дані до таблиці Google. Якщо під час процесу виникають будь-які винятки, функція виловлює їх і друкує відповідне повідомлення про помилку. Зокрема, він обробляє винятки `WorksheetNotFound` і `GoogleAuthError`, а також загальний. Ця функція викликається в основній програмі Flask, коли користувач успішно входить у систему та надсилає дані свого робочого дня. Він гарантує, що дані користувача надійно зберігаються в таблиці Google.

```python
@app.route('/add', methods=['GET', 'POST'])
def add_user():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        hashed_password = generate_password_hash(password, method='pbkdf2:sha256')

        conn = sqlite3.connect('D:\\learning\\dyplom\\instance\\user.db')
        cursor = conn.cursor()
        cursor.execute('INSERT INTO Users (username, password) VALUES (?, ?)', (username,hashed_password))
        conn.commit()
        conn.close()

        return redirect(url_for('add_user'))

    return render_template('add_users.html')
```

Рисунок 3.8 – Фрагмент коду створення нових користувачів

Фрагмент коду зображений на рисунку 3.8 бробляє реєстрацію користувачів на сторінці, яка зображена на рисунку 3.6. Коли користувач надсилає форму з новим іменем користувача та паролем, програма хешує пароль за допомогою функції generate_password_hash із модуля werkzeug.security. Потім підключається до бази даних SQLite. Цей фрагмент коду демонструє, як безпечно зберігати паролі користувачів, хешуючи їх перед збереженням у базі даних.

```python
@app.route('/endWorkDay', methods=['GET', 'POST'])
def end_work_day():
    username_hello = ""
    count = None

    if request.method == 'POST':

        email = request.form.get('username')
        username_hello = email

        if 'statistic' in request.form:
            try:
                username, domain = email.split("@")
                month = request.form.get('month')
                year = request.form.get('year')


                worksheet = client.open("Workdays").worksheet(username)


                date_column = worksheet.col_values(worksheet.find("Дата").col)


                count = 0


                for date_value in date_column[1:]:

                    day, value_month, value_year = date_value.split(".")


                    if value_month == month and value_year == year:
                        count += 1

                return render_template('endWorkDay.html', username_hello=username_hello, count=count)

            except Exception as e:
                print(f"Помилка: {e}")
                return "Помилка під час отримання статистики. Будь ласка, спробуйте знову."

    return render_template("endWorkDay.html", username_hello=username_hello, count=count)
```

Рисунок 3.8 – Вивід статистики

Усередині функції маршруту код перевіряє, чи є метод запиту POST. Якщо так, він отримує електронний лист із даних форми. Потім електронний лист використовується для відображення персоналізованого привітання на веб-сторінці.

Проводиться перевірка, чи натиснуто кнопку «статистика» у даних форми. Якщо так, переходить до отримання місяця та року з даних форми. Потім розділяє електронну пошту на ім’я користувача та домен за допомогою «@», порівнює значення місяць і рік із вибраними місяцем і роком.

Після циклу повертає візуалізований шаблон зі змінними і, переданими як аргументи. Загалом, цей код обробляє отримання статистики робочого дня з Google Таблиць і відображає її на веб-сторінці для користувача.

Рисунок 3.9 – Деталі сервісного акаунта Google Sheet API

Це JSON файл і він є ключовим елементом для забезпечення безпеки та аутентифікації в системі Google. Він містить дані конфіденційності, такі як приватний ключ та ідентифікатор проекту, які необхідні для взаємодії з сервісами Google.

Ці дані дозволяють програмі підтримувати автентифікацію та авторизацію в Google Sheet API, а також забезпечувати доступ до різних функціональних можливостей.

```
app.config['SECRET_KEY'] = secrets.token_hex(16)

scope = ["https://spreadsheets.google.com/feeds", "https://www.googleapis.com/auth/drive"]
creds = ServiceAccountCredentials.from_json_keyfile_name('./credent.json', scope)
client = gspread.authorize(creds)
```

Рисунок 3.10 – Налаштування програм

Код зображений на рисунку 3.10 виконує важливе завдання з налаштування програми Flask і встановлення з'єднання з Google Sheets API. Спочатку встановлюється закритий ключ для програми, який забезпечує шифрування і захист для сеансових користувачів. Це важливий захід безпеки. Потім він використовує приватний ключ для створення безпечного маркера сеансу та виконання інших операцій шифрування.

Після завантаження облікових даних виконується автентифікація, щоб програма могла взаємодіяти з Google Sheets API. Після підтвердження програма має дозвіл виконувати такі операції, як читання, запис та зміна даних у таблицях Google. Цей процес забезпечує безпечну та ефективну взаємодію між додатком та API Google Sheets, дозволяючи вам маніпулювати даними користувача, зберігаючи при цьому високий рівень безпеки.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Охорона праці в секторі інформаційних технологій є актуальним питанням у сучасному суспільстві. Стрімкий розвиток технологій та збільшення використання комп'ютерів та іншого обладнання створили нові виклики та ризики для тих, хто працює в цій галузі. Цей розділ дипломної роботи присвячений детальному дослідженню проблем охорони праці в IT-секторі, виявленню їх причин та наслідків, а також пошуку ефективних заходів для покращення безпеки та охорони здоров'я працівників.

4.1 Аналіз стану і заходи поліпшення виробничої санітарії і гігієни праці

IT-сфера – це галузь праці, яка являє собою роботу з комп’ютерними пристроями, такими як персональний комп’ютер, ноутбук, смартфон, планшет тощо. То ж в першу чергу потрібно розглянути охорону праці з боку використання даних гаджетів. Працівник майже весь увесь свій робочий час проводить перед монітором чи екраном, сидячи на робочому місці в приміщенні. Такий принцип роботи викликає певні шкідливі чинники, що впливають на фізичне та ментальне здоров’я людини. Найбільш поширеними чинниками, що призводять до погіршення стану здоров’я працівника є:

- Погіршення зору;
- Порушення постави;
- Сидячий метод роботи;
- Довготривалі розумові навантаження;
- Стресові ситуації.

Якщо не запобігти цим чинникам, то в працівників можуть з’явитись професійні хвороби. Серед недугів, які айтішники вважають професійними хворобами,

чимало діагнозів. Зокрема: остеохондроз, ожиріння, біль у шиї та спині, сколіоз, міжхребцеві грижі, геморой та закрепи, зниження зору, проблеми із органами травлення та виразка шлунку, безсоння, мігрень, гіпертонія, тривожні розлади і депресія, невроз тощо.

На сьогоднішній день є чимало досліджень на тему проблем зі здоров'ям працівників в ІТ. Для запобігання будь-яких з перелічених синдромів чи захворювань існують певні комісії чи відповідальні люди, що допомагають передувати недугам на підприємствах чи компаніях. Процес охорони здоров'я відрізняється в залежності від компаній, проте існують загальні стандарти і правила. Наприклад, інструктажі по охороні праці, рекомендації щодо роботи за комп'ютером та планові перевірки персоналу.

Дані процеси позитивно впливають на робочий колектив і в реальному часі впроваджуються все новіші розробки і вдосконалення для збереження здоров'я. Наприклад, останнім часом набувають популярності спеціальні комп'ютерні окуляри, в яких лінза захищає око від синього спектру екрану. Це не виключає повноцінного псування зору, проте дозволяє очам менше напружуватись під час роботи перед монітором.

Окрім цього, потрібно знати правила поводження з електроприладами, та як діяти під час не штатної ситуації. Перед включенням електроприладу необхідно візуально перевірити електрошнур на наявність механічних порушень. Електроприлади повинні бути надійно заземлені згідно з правилами улаштування приладу. Забороняється працювати з електроприладами вологими руками. Не залишати електроприлад без нагляду на довгий час, після закінчення роботи перевірити, чи всі прилади вимкнені. При виявленні або виникненні несправності в електроприладі негайно викликати електрика, що обслуговує прилад. Категорично заборонено виконувати будь-які ремонтні роботи самостійно.

4.2 Обґрунтування організаційно-технічних рекомендацій з охорони праці

Другий аспект з яким стикається працівник в сфері інформаційних технологій – це його робоче місце та приміщення, в якому він працює. Безпека під час роботи є надважливою складовою, тому існують наступні норми та правила:

- Освітлення. Середня величина освітленості приміщення повинна коливатися в межах близько 800-100 люксів. Досягти даної норми можна як і за природного освітлення вдень, так і зі штучним – у вечірню пору. Рекомендується використовувати LED-лампи з денним світлом в якості штучного освітлення;
- Заземлення. Робоче приміщення повинно мати один або декілька заземлювачів, що гарантують вивід струму під час короткого замикання чи інших ситуацій, при яких з’являються різницеві струми;
- Блискавкозахист. Важлива сукупність технічних заходів і засобів для зменшення збитку при ударі блискавки в приміщення;
- Первинні засоби гасіння пожежі. Наявність засобів для гасіння пожеж, таких як різного виду вогнегасники. Кнопки для виклику пожежної бригади та увімкнення тривоги у приміщенні. Датчики задимлення, що сповіщають про задимлення у приміщенні;
- Вентиляція. Мінімальна кількість свіжого повітря становить приблизно 60 метрів кубічних на одну людину. Відштовхуючись від цього, в приміщенні повинні встановленні штучні вентиляційні системи, або відкритий доступ до вікон, які забезпечують природну вентиляцію приміщення.

4.3 Пожежна безпека

Пожежна безпека є надважливою складовою в організації робочого процесу в сфері ІТ. Так як ця галузь активно пов’язана з використання

електроприладів в роботі, то ризик пожежної небезпеки є досить великим. Найбільш поширеними приміщеннями в ІТ-сфері є офіси, тому даний розділ потрібно розглядати згідно правил пожежної безпеки в офісних приміщеннях.

Управління з питань надзвичайних ситуацій та цивільного захисту населення Львівської міської ради рекомендує для ознайомлення та виконання всіма працівниками правил пожежної безпеки, які перебувають у службових приміщеннях, а також обслуговуючим персоналом:

- меблі та обладнання необхідно розміщувати таким чином, щоб забезпечувався вільний евакуаційний прохід до дверей виходу з приміщення (завширшки не менше 1 м). Евакуаційні шляхи та виходи необхідно постійно утримувати вільними, нічим не захаращувати;

- електромережі, електроприлади і апаратуру експлуатувати тільки у справному стані з урахуванням вказівок та рекомендацій підприємств-виготовлювачів. У разі виявлення пошкоджень електромереж, вимикачів, розеток та інших електровиробів слід негайно вимкнути їх та вжити необхідних заходів щодо приведення в пожежобезпечний стан;

- документи, папір та інші горючі матеріали слід зберігати на відстані не менше 1 м від електрощитів; 0,5 м від електросвітильників; 0,6 м від сповіщувачів автоматичної пожежної сигналізації та 0,15 м від приладів центрального водяного опалення.

- засоби протипожежного захисту слід утримувати у справному стані.

Усі працівники повинні вміти користуватись наявними вогнегасниками, іншими первинними засобами пожежогасіння, знати місце їх знаходження.

Відстань від найбільш віддаленого місця приміщення до місця розташування вогнегасника не повинна перевищувати 20 м.

У службових приміщеннях не допускати:

- влаштування тимчасових електромереж;
- прокладання електричних проводів безпосередньо по горючій основі;
- експлуатація електроприладів, які мають механічні пошкодження;
- захаращування підступів до засобів пожежогасіння;

- куріння, використовування легкозаймистих рідин;
- проведення вогневих, зварювальних та інших робіт без спеціального дозволу;
- вмикання електронагрівальних приладів (чайників, кип'ятильників тощо) без негорючих підставок та в тих місцях, де їх використання не передбачено (або заборонено);
- захаращування шляхів евакуації та евакуаційних виходів.

Адміністрація повинна зобов'язати відповідального за протипожежний стан службових приміщень після закінчення роботи:

- оглядати приміщення, переконуватись у відсутності порушень, що можуть призвести до пожежі;
- перевіряти, щоб скрізь було вимкнене освітлення, електроживлення приладів та обладнання (за винятком електрообладнання, яке за вимогами технології повинно працювати цілодобово).

У разі, якщо пожежі не вдалось уникнути, необхідно:

- терміново повідомити пожежну охорону по телефону 101, вказати при цьому адресу, кількість поверхів, місце виникнення пожежі, наявність людей, своє прізвище;
- організувати евакуацію людей та матеріальних цінностей;
- повідомити про виникнення пожежі адміністрацію та чергового (за його наявності);
- вимкнути, у разі необхідності, струмоприймачі та вентиляцію;
- розпочати гасіння пожежі наявними первинними засобами пожежогасіння;
- організувати зустріч підрозділів пожежної охорони й надати їм консультаційну та іншу допомогу в процесі гасіння пожеж.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

Дослідження та розробка інформаційної системи обліку робочого часу для відділу КІТ Львівського національного університету природокористування спрямована на потреби управління та технічного забезпечення інформаційної інфраструктури університету. Розроблена система має потенціал для вдосконалення процесів управління ресурсами кафедри, скорочення витрат часу на адміністративні процедури та підвищення загальної ефективності університету.

Процес розробки системи передбачав аналіз існуючих систем обліку робочого часу. Це призвело до створення ефективної інформаційної системи, яка враховує специфіку та потреби відділу КІТ, забезпечуючи точне, достовірне та зручне відображення робочого часу співробітників.

Впровадження розробленої системи стане маленьким кроком до модернізації управління персоналом та оптимізації робочих процесів університету. Подальший розвиток цієї системи має потенціал для підвищення ефективності роботи відділу КІТ та сприятиме покращенню загального рівня обслуговування співробітників університету.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Коваленко В.В., Розробка інформаційних систем. – Київ: Наукова думка, 2020.
2. Brown T., Advanced CSS and HTML5. – London: Pearson, 2019.
3. Bittner, K., Spence, I. . Моделювання варіантів використання. Велика Британія: Addison Wesley.2003
4. Chaudhuri, A. B. Блок-схема та основи алгоритму: Мистецтво програмування. Німеччина: Mercury Learning and Information, (2020).
5. Кодекс законів про працю України від 10.12.1971. Глава 4 Робочий час
6. Developers.google.com: https://developers.google.com/sheets/api/quickstart/python, (дата звернення 20.02.2024)
7. docs.python.org: https://docs.python.org/3.12/tutorial/index.html, (дата звернення 23.02.2024)
8. Павленко В.В. Веб-інженерія. – 2020. – №3. – С. 31-39.
9. Грязнова В. О., Єфіменко С. В. Основи методології програмування. - К.: ВПЦ "Київський університет", 2010.
10. Технології створення програмних продуктів та інформаційних систем : навч. посібник / М. Ю. Карпенко, Н. О. Манакова, І. О. Гавриленко ; Харків : ХНУМГ ім. О. М. Бекетова, 2017.
11. Алгоритми і структура даних: Навчальний посібник / В.М.Ткачук. - ІваноФранківськ : Видавництво Прикарпатського національного університету імені Василя Стефаника, 2016.
12. Dev.ua. URL: https://dev.ua/news/aitishnyky-i-khvoroby-1661439760 (дата звернення 23.04.2024). 52
13. Львівська міська рада. URL: https://city-adm.lviv.ua/lmr/socialniiniciativi/2097-zakhody-pozhezhnoi-bezpeky-u-sluzhbovykh-prymishchenniakhofisakh (дата звернення 24.05.2023).
14. Тимочко В.О., Городецький І.М., Березовецький А.П., Мазур І.Б. та ін. Безпека життєдіяльності та охорона праці. Навч. посібник. Львів: Сполом. 2022.

15. Пістун І. П., Березовецький А. П., Тимочко В. О., Городецький І. М. Охорона праці (гігієна праці та виробнича санітарія): навч. посіб. / за ред. І.П.Пістуна. Львів: Тріада плюс, 2017. Ч. І. 620 с.