ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА ДЕСКТОПНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗБЕРІГАННЯ ЗДОРОВ'Я ОЧЕЙ ПІД ЧАС РОБОТИ ЗА КОМП'ЮТЕРОМ НА МОВІ C#»**

Виконав: студент 4 курсу, групи ПД–41
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності/спеціалізації)

Дібрій Д.А.
(прізвище та ініціали)

Керівник Трінтіна Н.А.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Київ –2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти -«Бакалавр»
Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Негоденко О.В.
"____" _____2021 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТА

Дібрію Данилу Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка досконалого програмного забезпечення для зберігання здоров'я очей під час роботи за комп'ютером на мові C#»

Керівник роботи: Трінтіна Н.А., к.т.н., доцент кафедри ІПЗ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «12» березня 2021 року №65.

2. Строк подання студентом роботи «1» червня 2021 року

3. Вхідні дані до роботи
Методи збереження здоров'я очей під час роботи за комп'ютером;
Науково-технічна література з питань, пов'язаних з розробкою програмного забезпечення;

4. Зміст розрахунково-пояснювальної записки(перелік питань, які потрібно розробити).

4.1 Методи збереження здоров’я очей користувача під час роботи за комп’ютером
4.2 Аналіз використаних технологій.
4.3 Опис проектування системи.
4.4 Опис використаних технологій.

5. Перелік демонстраційного матеріалу (назва основних слайдів)
1. Актуальність проблеми
2. Аналоги
3. Технічне завдання
4. Використані технології
5. Принцип взаємодії користувача та системи
6. Висновки

7. Дата видачі завдання «19» квітня 2021

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	19.04.2021	Виконано
2	Вимоги до системи	20.04-21.04	Виконано
3	Аналіз технологій	21.04-22.04	Виконано
4	Проектування системи	22.04-24.04	Виконано
5	Розробка програмного забезпечення	24.04-26.04	Виконано
6	Вступ, висновки, реферат	27.04-28.04	Виконано
7	Розробка обов‘язкових демонстраційних матеріалів	28.04.2021	Виконано
8	Попередній захист роботи		
9	Здача роботи	1.06.2021	

Студент _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи 48 с., 44 рис., 14 джерел.

МЕТОДИ ЗБЕРІГАННЯ ЗДОРОВ'Я ОЧЕЙ, ПРОЕКТУВАННЯ СИСТЕМИ, ПРОЦЕС РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, API WINDOWS FORMS, ДЕСКТОПНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об'єкт дослідження – методи зберігання здоров'я очей під час роботи за комп'ютером.

Предмет дослідження – користувацьке програмне забезпечення для зберігання здоров'я очей користувача відповідно до вимог українського законодавства та сучасних офтальмологічних досліджень.

Мета роботи – розробити десктопне програмне забезпечення під назвою «Watch Your Eye» для зберігання здоров'я очей користувача під час роботи за комп'ютером за допомогою мови C#.

Методи дослідження – у роботі були використані теоретичні методи дослідження для аналізу інформації отриманої з науково-технічної літератури, емпіричні методи для нагляду, порівняння та постановки експериментів.

Через стрімкий розвиток інформаційних технологій все більше людей починає проводити більше чотирьох годин на день за екранами комп'ютерів, що призводить до появи професійних захворювань. Одним з цих захворювань є комп'ютерний зоровий синдром, симптоми якого ведуть до погіршення зору.

Проблемою існуючого програмного забезпечення яке допомагає у профілактиці синдрому є недостатня інформативність та обмежений функціонал, який не дає користувачу комфортно займатись профілактичними вправами.

Особливістю створеного додатку є можливість простого налагодження часу роботи, часу відпочинку, вправи для очей та можливість перегляду користувацької статистики при відсутності необхідності з'єднання з інтернетом, тобто враховані та виправлені недоліки вже існуючих програм, що робить його значно у очах користувача і надає повний функціонал для профілактики.

При виконанні роботи була спроектована система роботи додатку, проаналізовані моделі розробки ПЗ, середовища розробки, інтерфейсів програмування додатків для вибору того що найбільш відповідає вимогам для розробки додатку. Таким чином було прийняте рішення про розробку програми у середовищі розробки Microsoft Visual Studio за допомогою інтерфейсів програмування додатків Windows Forms на мові програмування C#.

Отже, розроблено та описано десктопне програмне забезпечення, завданням якого є забезпечення для зберігання здоров’я очей під час роботи за комп’ютером.

Дану програму може бути використано на підприємствах які прагнуть зберегти здоров’я своїх працівників, або для домашнього персонального використання.

Галузь використання – системи захисту здоров'я користувачів під час роботи з персональними комп'ютерами.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

КЗС(CVS) – комп’ютерний зоровий синдром.

ПЗ – програмне забезпечення.

API – інтерфейс програмування додатків.

IDE – інтегроване середовище розробки.

БД – база даних.

ВСТУП

Обґрунтування вибору теми та її актуальність: активний розвиток інформаційних технологій призвів до необхідності використання комп'ютерів майже у всіх сферах праці. І хоча автоматизація ряду робочих процесів призвела до значного підвищення ефективності підприємств, щоденна восьми годинна робота призводить до появи нових професійних захворювань, одне з них – це комп'ютерний зоровий синдром, або КЗС. КЗС виникає під час довгої безперервної роботи за екраном монітора та є одним з чинників погіршення зору користувача.

Методи профілактики КЗС вимагають регулярних перерв та спеціальних вправ для очних м'язів, але монотонність та складність постійного слідкування за часом під час роботи призводить до того що користувачі просто не виконують необхідної профілактики. Через це виникає необхідність створення користувацького програмного забезпечення яке прийме відповідні задачі та дозволить зменшити негативний вплив довгої роботи за персональним комп'ютером.

Ступінь вивчення проблеми: на сьогоднішній день існує ряд програм які допомагають у профілактиці здоров'я зору, але усі вони бракують ряду необхідних функцій. Так продукт «EyeCare» вимагає постійного інтернет з'єднання, «BreakTime» не має вбудованих вправ, а «EyeLeo» має застарілий інтерфейс та не надає користувачу статистики про час роботи та відпочинку.

Через існуючі недоліки ці програми не дозволяють повністю автоматизувати профілактичну роботу користувача, що призводить до перекладання частини цієї роботи на користувача та зменшує привабливість продукту.

Об'єктом дослідження є методи зберігання здоров'я очей під час роботи за комп'ютером.

Предметом роботи є десктопне користувацьке програмне забезпечення(ПЗ) для зберігання здоров’я очей користувача відповідно до вимог українського законодавства та сучасних офтальмологічних досліджень.

Метою роботи є створення десктопного користувацького ПЗ для зберігання здоров’я очей користувача під час роботи за комп’ютером на мові C#.

за допомогою створення користувацького ПЗ розробленого на мові C#.

Завдання роботи: дослідити методи збереження здоров’я очей користувача під час роботи за комп’ютером, проаналізувати принцип роботи і виявити недоліки існуючих систем по збереженню зору, дослідити основні методології розробки ПЗ, дослідити необхідні інструменти для розробки ПЗ, спроектувати сисему,розробити ПЗ для збереження здоров’я очей, провести тестування створеного ПЗ.

Враховуючи специфіку програмного продукту, найкращим рішенням є прикладне користувацьке програмне забезпечення на платформі Windows з подієво-орієнтованою архітектурою, де для реалізації інтерфейсу користувача буде використано API Windows Forms, а для збереження даних користувача технологія ADO.NET. Такий тип продукту дозволить охопити максимальну аудиторію, оскільки більша частина користувачів ПК працює саме на платформі Windows, а технологія ADO.NET надає можливість для подальшого масштабування горизонтального масштабування системи.

Практична значущість результатів: даний продукт може бути використаний на підприєствах у яких значна частина працівників працює за стаціонарними компьютерами більше чотирьох годин на день, так допоможе зберегти їх зір. Таким чином роботодавець отримає здорового працівника єкий зможе ефективно виконувати свої обов'язки, а працівник збереже свій зір. Також програма може бути використана у сфері домашнього застосування, оскільки ПК є не тільки інструментом праці, а й надає розважальний контент від якого буває важко відірватись. Створене ПЗ допоможе таким людям зробити перерву під час перегляду розважального контенту та зберегти здоров'я свого зору.

1 АНАЛІЗ МЕТОДІВ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗБЕРІГАННЯ ЗДОРОВ'Я ОЧЕЙ

1.1 Методи збереження здоров'я очей під час роботи за комп'ютером

Згідно дослідження академічного журналу «Medical Practice and Reviews» [1], 70% людей які постійно працюють за комп'ютером страждають від тих чи інших проблем із зором. Відсоток студентів інженерних спеціальностей навіть більше, він становить 81.9%.

Найчастіше проблеми починаються з комп'ютерного зорового синдрому, або КЗС. КЗС виникає у людей які працюють за екранними пристроями на постійній основі більше чотирьох годи на добу, найчастішими симптомами КЗС є:

- головні болі
- почервоніння та свербіж очей
- напруга очей
- замилення зору
- сухість очного яблука

І хоча сам КЗС не впливає на зір, постійний вплив його симптомів призводить до порушень зору.

Сама по собі профілактика КЗС не вимагає стаціонарного лікування або вживання медичних препаратів, згідно офтальмологічних досліджень для зняття ефекту КЗС потрібно дотримуватись досить простих методів профілактики:

- кожні 20 хвилин дивитись на віддалені предмети протягом 20 секунд;
- робити перерви регулярність та час яких встановлюється в залежності від часу роботи за екранними пристроями;
- знімати напругу з очних м'язів за допомогою спеціальних вправ;
- тримати робоче місце відповідно відповідності з встановленими санітарними нормами;

Важливо відзначити, що дотримання профілактичних заходів не призводить до покращення зору у разі його погіршення, а слугує лише у якості профілактики для запобігання погіршення його поточного стану за допомогою зменшення загальної напруги на очі [2].

Профілактичні дії знімають ефект КЗС, але частково через недостатню інформованість людей, частково через те що подібні методи вкючають в себе доволі монотонні та репетативні задачі планування яких відволікає від основної цілі роботи за комп'ютером, користувачі просто не проводять яких би то ні було профілактичних дій. Так за даними МООЗ 2,2 мільярди людей страждають порушеннями зору, значна кількість цих людей могла уникнути їх виконуючи профілактику [3].

1.2 Аналіз вимог українського законодавства щодо правил роботи з візуальними дисплейними терміналами

Слід розглянути не тільки офтальмологічні дослідження, а й законодавстві норми щодо правил роботи, оскільки в них також містяться рекомендації щодо праці з дисплеями комп’ютерів.

Згідно постанови Міністерства охорони здоров'я України, для люди які працюють протягом дня більше 50% часу робочої зміни з дисплеями, мають передбачатися внутрішньо змінні режими праці і відпочинку [4].

З цього можна зробити висновок що основною діяльністю працівника потрібно вважати таку, що становить не менше чотирьох годин на день.

Екрані пристрої — електронні засоби для відтворення будь-якої графічної або алфавітно-цифрової інформації такі як монітори та інші новітні розробки у сфері інформаційних технологій.

Згідно наказу Міністерства соціальної політики України щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями, вимоги не поширюються на:

- робочі місця здобувачів освіти у комп’ютерних класах закладів освіти, тобто школярів та студентів;
- робочі місця пілотів, водіїв або операторів транспортних засобів, обладнані екранними пристроями у системах оброблення даних на борту засобів сполучення, екранні пристрої у складі машин і обладнання, що переміщуються в процесі роботи;
- робочі місця працівників, які займаються обслуговуванням, ремонтом і налагодженням екранних пристроїв;
- портативні системи оброблення даних, якщо вони не постійно використовуються на робочому місці;
- робочі місця працівників які працюють за касовими апаратами та приладами з невеликими пристроями індикації даних або результатів вимірювання;
- цифрові друкувальні машини, обладнані візуальними дисплейними терміналами (дисплейні друкувальні машини);
- планшети, смартфони, мобільні телефони.

Ознайомившись наказом з Міністерством охорони здоров'я України “Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями” було встановлено, що працівникам які працюють з візуальними дисплейними терміналами електронно-обчислювальних машин при 8-годинній денній робочій зміні встановлюються наступні перерви в залежності від характеру праці:

- для розробників програм із застосуванням ЕОМ перерва для відпочинку становить 15 хвилин через кожну годину;

- для операторів із застосування ЕОМ регламентовані перерви для відпочинку тривалістю 15 хвилин через кожні дві години;

- для операторів комп'ютерного набору перерви становлять 10 хвилин після кожною години роботи за ВДТ;

- коли виробничі обставини не дозволяють по тим чи іншим причинам застосувати регламентовані перерви, тривалість безперервної роботи з ВДТ не повинна перевищувати 4 години.

Також роботодавець повинен проінформувати працівників під розписку про умови праці та наявність шкідливих виробничих факторів, які виникають під час роботи з екранними пристроями та ще не усунуто, а також про можливі наслідки їх впливу на здоров'я працівників.

У разі невиконання порушення цих норм роботодавець повинен виплатити штраф, максимальний розмір якого його не може перевищувати 5% середньомісячного фонду зарплати за попередній рік.

Але через складність відстеження подібних вимог зазвичай роботодавці уникають виконання цього закону, та й працівники через монотонність і репетативність профілактичних методів та недостатню інформованість не виконують профілактичних вправ.

1.3 Аналіз існуючого програмного забезпечння для зберігання здоров'я очей під час роботи за комп'ютером

«EyeCare» це безкоштовне розширення для браузерів Google Chrome ціллю якого є захист зору користувача за правилом 20-20-20.

Правило 20-20-20 — кожні 20 хвилин дивитися на предмет розташований на дистанції 20 футів та більше протягом 20 секунд. Правило базується на дослідженні американської оптометричної асоціації [5].

«EyeCare» надає користувачу можливість встановити регулярність перерв, демонструє вправи для зняття напруги з очних м'язів, та надсилає користувачу повідомлення про необхідність перерви. (Рисунок 1.1).

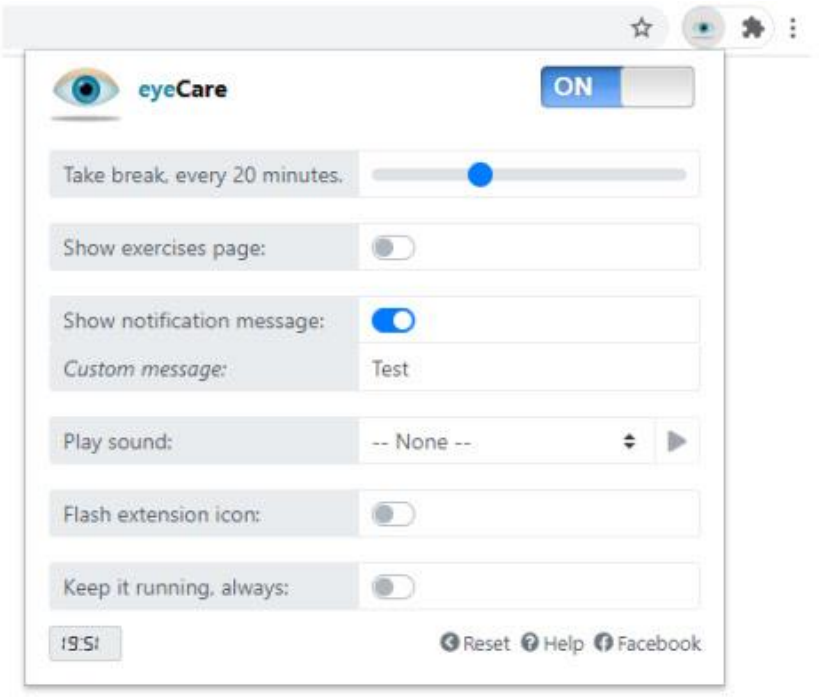


Рисунок 1.1 – Розширення «EyeCare»

Переваги:

- Можливість встановити час роботи.
- Демонструє вправи для очних м’язів.
- Надає повідомлення про початок перерви.
- Повністю безкоштовний.

При цьому це розширення має ряд серйозних недоліків, а саме:

- Неможливість встановити час відпочику при можливості збільшити час роботи не тільки робить правило 20-20-20 неефективним, а й не надає користувачу достатньої гнучкості. Так eyeCare не зможуть використовувати ряд користувачів яким необхідні довші перерви.
- Необхідність постійного підключення до інтернету. У разі відсутності інтернету користувач не зможе користуватись розширенням.
- Відсутність користувальницької статистики. Користувач не має можливості оцінити час витрачений на роботу та відпочинок.

«BreakTime» це платний додаток розроблений для операційної системи macOS яку було розроблено у 2011, яка призначена для планування перерв під час роботи.

Він надає користувачу можливість самостійно обирати час роботи і відпочинку та надає можливість вмикати або вимикати повідомлення про необхідність відпочинку. Повідомлення супроводжуються досить гучними звуками, це зроблено для того щоб користувач точно не пропустив перерву. (Рисунок 1.2).

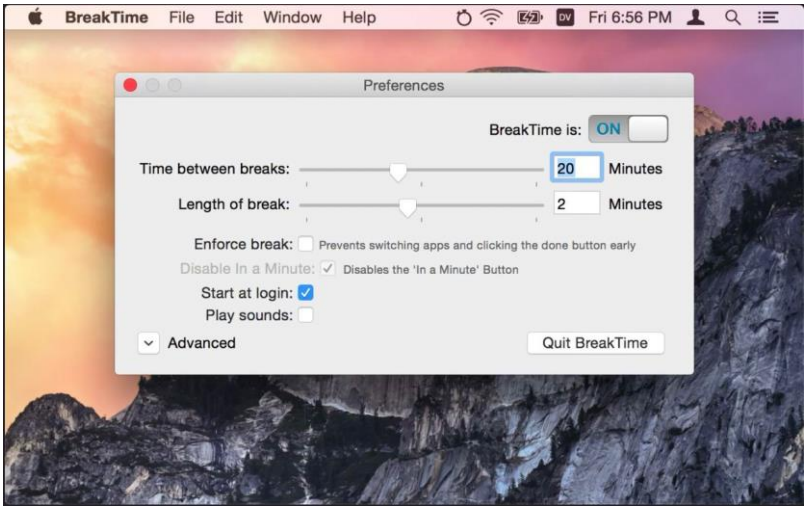


Рисунок 1.2 – Утиліта «BreakTime»

Головною проблемою цього додатку є повна відсутність вправ для очей. А зважаючи на те що вона не є безкоштовною та розповсюджується лише для операційної системи macOS, яку використовує досить невелика кількість користувачів, зацікавленість користувача у експлуатації такого ПЗ є мінімальною.

Переваги:

- Можливість гнучкого налаштування часу роботи та часу перерв.
- Наявність повідомлень та можливість їх налаштування.
- Вістутність потреби у постійному з'єднанні з мережею інтернет.
- Візуально привабливий інтерфейс.

Недоліки:

- Платна.
- Не містить вправ для очей.
- Працює лише з операційною системою macOS, яка займає досить невелику долю ринку настільних ПК.
- Користувачі скаржаться на присутність помилок які заважають роботі [6].
- Відсутність статистики роботи та відпочинку.

EyeLeo це безкоштовний додаток який було розроблено у 2010 році та працює на операційних системах сімейства Windows, що нагадує робити регулярні перерви під час роботи за комп'ютером.

Він надає можливість і налагоджувати час роботи та відпочинку, надає користувачу повідомлення, і має вбудовані вправи для очей, які демонструє маскот «EyeLeo» – леопард.

Хоча «EyeLeo» та «BreakTime» не є конкурентами, оскільки розроблялись для різних операційних систем, не можна не відзначити те, що EyeLeo в порівнянні з «BreakTime» надає набагато більший функціонал будучи повністю безкоштовним, хоча і має дещо застарілий зовнішній вигляд.(Рисунок 1.3).

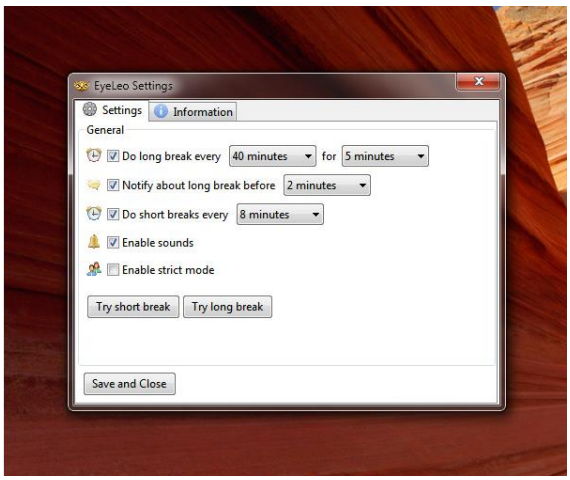


Рисунок 1.3 – Додаток «EyeLeo»

Значним недоліком є неможливість відключення повідомлень з вправами, у разі якщо користувач не бажає бачити випливаючі вікна які займають простір на екрані, він може тільки зупинити роботу програми. (Рисунок 1.4).

Рисунок 1.4 – Повідомлення від «EyeLeo»

Як і у «BreakTime», в «EyeLeo» повністю відсутня будь-яка статистика про загальний час роботи та відпочинку користувача. Також можна відзначити дещо застарілий інтерфейс додатку, оскільки він був розроблений ще у 2010 році для операційної системи Windows 7.

Перваги:

- Безкоштовна
- Можливість гнучкого налаштування часу роботи та часу перерв.
- Наявність повідомлень з вбудованими вправами.
- Вістутність потреби у постійному з'єднанні з мережею інтернет.

Недоліки:

- Неможливість вимикання повідомлень з вправами які зпливають на екрані користувача.
- Відсутність статистики використання роботи та відпочинку користувачем.

Висновок розділу

Після аналізу існуючого ПЗ та методів захисту здоров'я очей під час роботи за комп'ютером, можна зробити висновок про недостатню його ефективність через відсутність імплементації ряду необхідних функцій для профілактики здоров'я очей.

Через це виникає необхідність у створенні достконалого ПЗ, яке буде позбавлене усіх недоліків конкурентів і зможе допомагати користувачу дотримуватись необхідних профілактичних методів.

2 АНАЛІЗ ТЕХНОЛОГІЙ ТА ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Вибір моделі розробки ПЗ та створення вимог

Перш ніж почати проектвати сисему необхідно визначитмсь, за якою саме моделлю розробки буде створено ПЗ. Вибір моделі сер'йозно впливає на розробку, визначаючи розклад та стратегію.

Модель розробки ПЗ – це структура яка систематизує різні види проектної діяльності та їх взаємодію. Вибір тієї чи іншої моделі залежить від масштабу і складності проекту.

Найвідомішими моделями розробки є водоспадна, ітераційна, спіральна та гнучка. Проведемо аналіз кожної цієї моделі для визначення найбільш ефективної для розробки ПЗ.

Водоспадна модель передбачає одниничне виконання кожної фази проекту. Ця модель складається з таких фаз розробки: створення вимог, створення архітектури та дизайну, написання коду, тесування, розгортання проекту, подальша підтримка. (Рисунок 2.1).

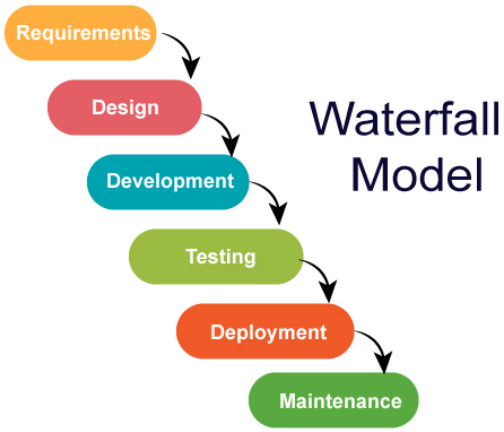


Рисунок 2.1 – Водоспадна модель

Наступна фаза проекту може початись лише після повного закінчення попередньої. Це породжує багато проблем, оскільки при розробці ПЗ часто необхідно повертатись до попередніх фаз, щоб виправити певні помилки. З точки зору тестування ця модель погана тим, що тестування з'являється тут лише з середини розвитку проекту.

Через ці недоліки водоспадна модель практично не використовується для побудови користувацького ПЗ, за винятком проектів в яких вимоги дуже стабільні, наприклад створення ПЗ для будування космічних ракет.

Ітераційна модель – відповідно до назви, ця модель бачить створення ПЗ за допомогою певних циклів розробки які повторюються, при цьому з кожним циклом збільшується відповідність продукту до вимог. Після загального планування, починається фаза планування поточної ітерації та розробляються відповідні вимоги, проводиться аналіз та імплементація, тестування та оцінка результату. Довжина ітерацій може змінюватись в залежності від різних факторів які пов’язані з розробкою. (Рисунок 2.2).

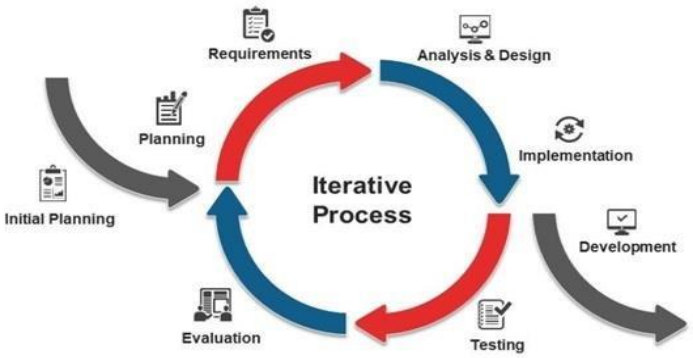


Рисунок 2.2 – Ітераційна модель

Ітераційна модель дуже добре працює при побудові великих і складних проектів, які виконуються великими командами протягом тривалих термінів. Однак у разі створення невеликих програм загальна громіздкість моделі буде скоріше заважати розробці ніж допомагати.

Спіральна модель являє собою окремий випадок ітераційної моделі, в якому особлива увага приділяється управлінню ризиками. Вона має наступні фази: встановлення цілей, аналіз ризиків, розробка проміжної версії, планування наступного циклу. (Рисунок 2.3).

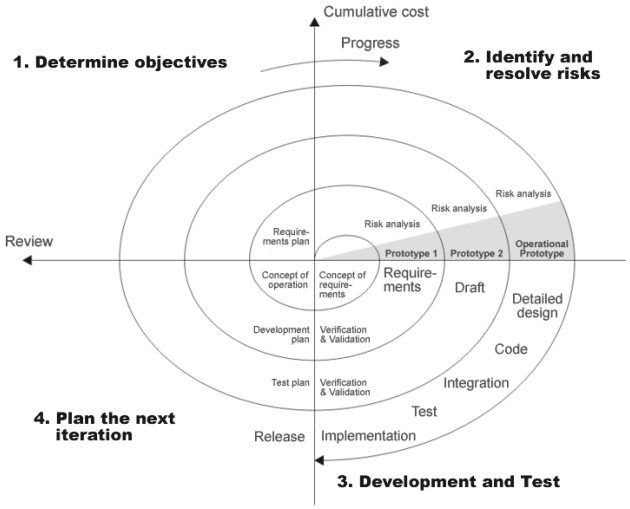


Рисунок 2.3 – Спіральна модель

З точки зору тестування підвищена увага до ризиків є відчутною перевагою при використанні спіральної моделі для розробки проектів, в яких вимоги

природним чином є складними і вимоги яких можуть змінюватись з часом. Але як ітераційна модель не дуже підходить для створення невеликих програм.

Гнучка модель – це модель яка вбирає у себе різні підходи розробки ПЗ і базується на «Agile-маніфесті».

Правила цього маніфесту наступні:

- Люди і їх взаємодія важливіше процесів та інструментів.
- Працюючий продукт важливіше вичерпної документації.
- Співпраця з замовником важливіше узгодження умов контракту.
- Готовність до змін важливіше проходження попереднім планом.

В основі гнучкої моделі лежать дорозвинуті підходи водоспадної, итерационной ітераційної та спіральної моделі, причому значно знижується бюрократичної складова розробки. Ця модель надає значно зменшує процес змін вимог та їх адаптпцію у робоче ПЗ. Як і в ітераційній моделі, проект розділено на певні ітерації. Кожна ітерація має в собі наступні фази: планування цілі ітерації, планування задач які потрібно виконати, тестування, та оціка результатів роботи виконаній у цій ітерації. У гнучкій моделі кожна ітерація триває протягом двох неділь та називается «Sprint». (Рисунок 2.4).

Рисунок 2.4 – Гнучка модель

Можна сказати, що гнучка модель це полегшена з точки зору документації суміш итерационої та спіральної моделей. Така модель широко використовується для побудови невеликих проектів вимоги яких можуть швидко змівнитись.

Головним недоліком гнучкої моделі є складність її застосування у великих проектах. Оскільки документація виносится на другий план це призводить до впровадження невірних функцій через недостанє розуміння цілей та задач проекту.

Приймаючи до уваги усі переваги та недоліки описаних вище моделей, було прийнято рішення про використання гнучкої моделі розробки, оскільки створюване ПЗ невелике за масштабами і занадто велика документація лише сповільнить розробку, не надавши значних переваг.

Навіть для гнучких моделей розробки ПЗ необхідно мати базову документацію, оскільки знання необхідних функцій проекту допоможе у подальшому виборі технологій для їх реалізації. Приймаючи це до уваги, було прийнято рішення про необхідність створення функціональних та не функціональних вимог продукту.

Проаналізувавши існуюче програмне забезпечення для захисту здоров’я очей, законодавство України про охорону праці під час роботи з екранами комп’ютерів та профілактичні методи, можна створити список з необхідних вимог до ПЗ.

Вимога до ПЗ – це певний опис того, які функції необхідні бути присутні в додатку та як саме вони повинні виконуватись в процесі вирішення задачі користувача.

Вимоги дозволяють визначити що саме потрібно проектувати, реалізовувати і тестувати, тому є початковою і дуже важливою частиною проектування системи. Це пов’язано з тим, що при відсутності або неправильному формуванні вимог ПЗ на виході може реалізовувати поставлено задачу не правильно, оскільки вона не було точно сформована. Тобто колосальна робота буде виконана даремно та повинна початись наново, що призводить до значних втрат часу та грошей. (Рисунок 2.5).

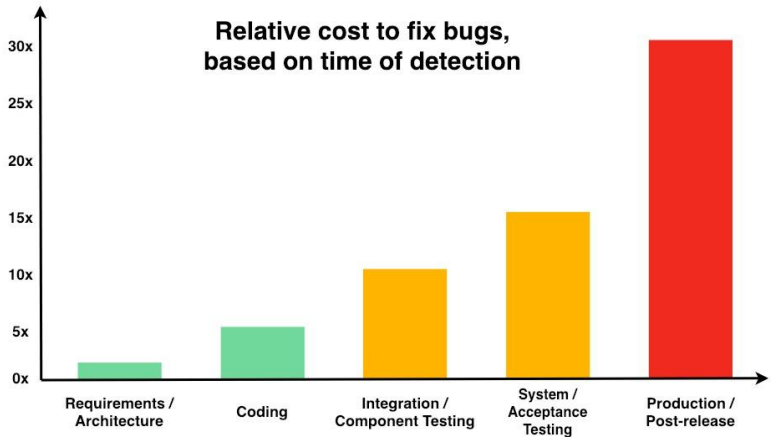


Рисунок 2.5 – Вартість виправлення помилки на різних стадіях проекту

Правильно складені вимоги дозволяють:

- Зрозуміти, що і як саме система повинна робити.
- Надають можливість оцінити зміни в проекті.
- Є основою для формування плану проекту.
- Допомагають запобігати або вирішувати конфліктні ситуації.
- Спрощують розстановку пріоритетів при обиранні задач завдань.
- Дозволяють оцінити ступінь прогресу в розробці проекту.

Існують різні методи створення вимог, такі як інтервью з замовником, анкетування, самостійний опис, мозговий штурм, аналіз документів, спостереження. Через те що у попередньому розділі дані були отримані з досліджень в офтальмології, державних законів України та аналіз схожого ПЗ методами створення вимог було обрано аналіз документів та спостереження.

Вимоги до ПЗ можуть бути функціональними та не функціональними в залежності від поставленої до них задачі.

Функціональні вимоги – це вимоги які описують поведінку системи, точніше її дії (обчислення, перевірки, обробку даних). Варто пам'ятати, що до поведінки системи відноситься не тільки те, що система повинна робити, але і те, що вона не повинна робити.

Нефункціональні вимоги – це вимоги що описують властивості системи (зручність використання, надійність).

Приймаючи це до уваги, були складені відповідні функціональні та нефункціональні вимоги.

Функціональні вимоги:

- Система повинна надавати користувачу можливість самостійно обрати час роботи і відпочинку згідно з значеннями заснованими на правилі 20-20-20 та вимог українського законодавства щодо правил роботи з візуальними дисплейними терміналами. Створемо таблицю цих значень:

Таблиця 2.1 Необхідні роботи і відпочинку

Час роботи	20хв	1год	2год
Час відпочинку	20сек	10хв	15хв

- У разі початку перерви ПЗ повинно надавати користувачу вправи для зняття напруги з очей такі як рух очей по колу, блимання очами, рух очей у різні сторони, спостерігання віддалених предметів.
- У разі початку перерви ПЗ повинно надавати користувачу можливість скасувати перерву.
- ПЗ повинно надсилати користувачу повідомлення про необхідність перерви або повернення до роботи якщо відповідна функція увімкнена користувачем.
- ПЗ повинно формувати недільну статистику про час проведений за роботою і час проведений за відпочинком для самостійного відстежування та зберігати її у базі даних.
- При початку нової неділі ПЗ повинно видаляти дані про стару.
- В процесі інсталяції на ПК користувача інсталятор має перевіряти залишок вільного місця на цільовому носії.
- ПЗ повинно автоматично зберігати дані про час проведений за роботою та відпочинком.

- ПЗ повинно виконувати усі свої функціональні задачі незалежно від того чи є у користувача інтернет.
- ПЗ повинно зберігати дані БД протягом тижня у якому БД була створена.
- Файли БД повинні зберігатись у окремій папці в калозі розміщення програми.

Нефункціональні вимоги:

- ПЗ не повинно використовувати більше 5% ресурсів оперативної пам'яті, процесору та відеокарти системи.
- ПЗ повинно надавати користувачу можливість зміни візуального оформлення інтерфейсу, а саме можливість перемикати інтерфейс у білу або чорну тему.
- ПЗ повинно розроблене для операційної системи яка займає найбільшу долю ринку стаціонарних комп'ютерів протягом останніх трьох років.
- Шрифти тексу інтерфейсу програми повинні бути типу “Century Gothic”.
- Загальна кольорова гамма світлої теми інтерфейсу ПЗ повинна бути виконана у блакитних тонах.
- Загальна кольорова гамма темної теми інтерфейсу ПЗ повинна бути виконана у темно-зелених тонах.
- ПЗ повинно надавати користувачу коротку довідку про КЗС.
- Час перерви і відпочинку повинен збігатись з встановленим в налаштуваннях ПЗ. ПЗ не повинно зупиняти роботу у разі згортання вікна.
- ПЗ не повинно демонструвати користувачу вправи в робочий час.
- ПЗ не повинно самостійно розгортатись якщо до цього було згорнуто користувачем.

2.2 Вибір інструментарію для побудови системи

Одною з необхідних функціональних вимог є охопити якомога більшої кількість користувачів які працюють за екраними пристроями, тому слід визначити для якої саме операційної системи потрібно створювати ПЗ.

Найкращим способом дізнатись про долю користувачів тієї чи іншої операційної системи буде перевірення статистичних даних про частину ринку яку займає та чи інша система.

Так за даними компанії GlobalStats доля операційних систем сімейства Windows за останній рік стабільно складає 75-76% по всьому світу, а доля на ринці україни навіть більша—85% [7]. (Рисунок 2.6).

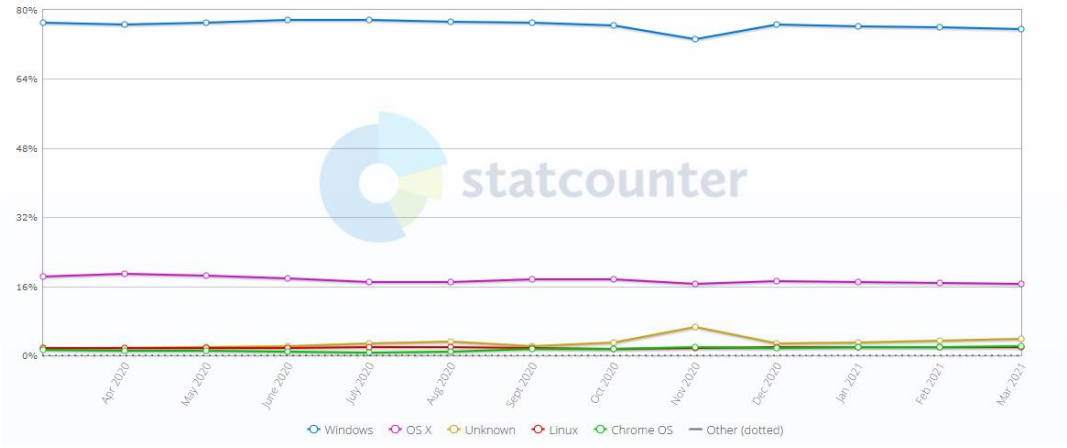


Рисунок 2.6 – Рівень охоплення долі ринку різних операційних систем

Тож доречно буде обрати саме операційну сімейства Windows.

Windows – це сімейство операційних систем (ОС) від корпорації Microsoft, яке надає користувача можливість працювати з файлами, заупскати програмне забезпечення та підключатись до мережі інтернет за допомогою графічного інтерфейсу.

Завдяки постійній підтримці від компанії Microsoft, програми створені для старіших членів сімейства Windows можуть запускатися на новітніх системах, так програма розроблена для Windows 7 може будти використана у системах Windows

8 та Windows 10. Останньою на сьогоднішній момент операційною системою є Windows 10, тому саме її буде використано.

Одним з найважливіших елементів в процесі розробки програми є вибір правильної IDE, що залежить не тільки від платформи, але і рівня власної підготовки.

Інтегроване середовище розробки (IDE) – це комплекс програмних засобів, який використовується програмістами для розробки програмного забезпечення. Саме у ньому програміст пише код, ловить помилки і спостерігає результат.

Розглянемо популярні середи розробки:

IntelliJ IDEA – це середовище розробки програмного забезпечення для багатьох мов програмування, основними з яких є Java, JavaScript та Python.

Мови Java та JavaScript які підтримує IntelliJ IDEA найчастіше використовують у роботі з веб-додатками, а Python – для машинне навчання. Оскільки розробляється система яка не буде взаємодыяти з мережею інтернет та у який не буде використовуватись машине навчання, ця середа розробки не підійде.

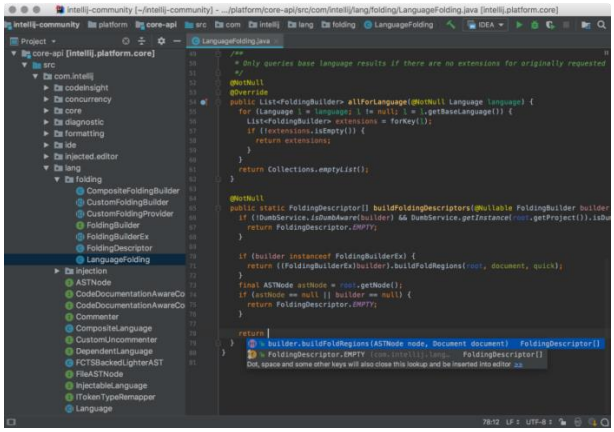


Рисунок 2.7 Інтерфейс IntelliJ IDEA

Xcode – це середовище розробки програмного забезпечення для платформ macOS, iOS, watchOS і tvOS, розроблена корпорацією Apple. Xcode підтримує мови C, C ++, Objective-C, Objective-C ++, Swift, Java, AppleScript, Python і Ruby. Що надає великий простір для розробки додатків різних видів, але оскільки це

IDE в першу чергу орієнтовано на розробку для операційних систем macOS, воно не підходить, оскільки було вирішено створювати програму для сисеми Windows.

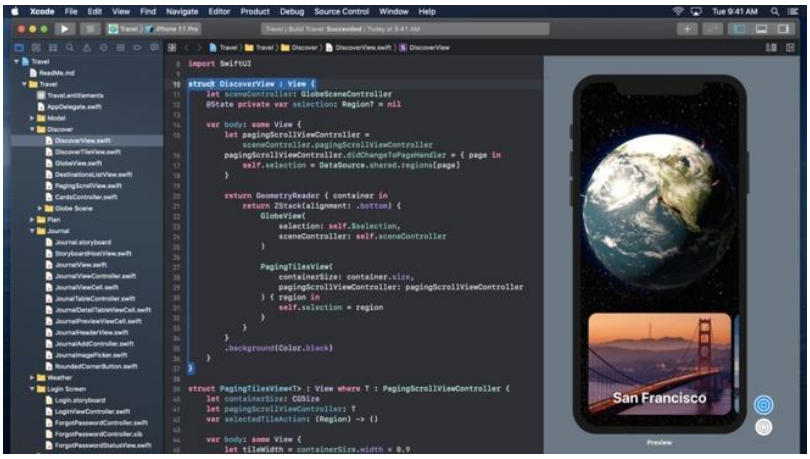


Рисунок 2.8 Інтерфейс Xcode

Visual Studio – це повнофункціональне інтегроване середовище розробки для написання, налагодження, тестування і розгортання коду на мовах C#, C++, Visual Basic .NET, Visual J Sharp.

Visual Studio була розроблена і підтримується компанією Microsoft, та надає багатий інструментарій для створення графічних додатків саме для платформи Windows. Також ця IDE включає в себе дизайнер класів і дизайнер схеми бази даних.

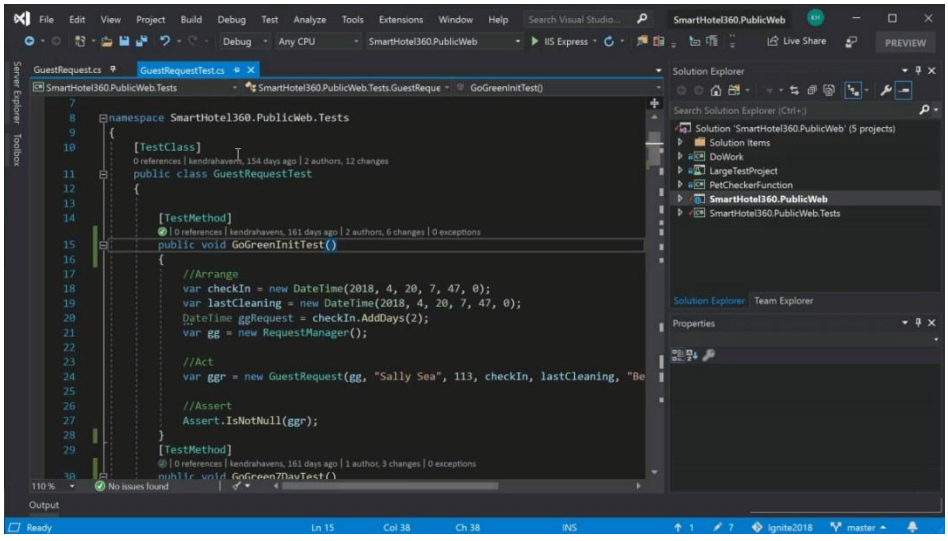


Рисунок 2.7 Інтерфейс Visual Studio

Беручи до уваги можливості які надає Visual Studio для створення графічних додатків для операційної системи Windows, саме цю IDE обрано для розробки.
З списку наданих цією IDE мов програмування обрано об'єктно-орієнтовану мову C#, так як я маю досвід розробки на цій мові.

Оскільки ПЗ буде представляти собою графічний інтерфейс який буде реагувати на дії користувача, найкращим рішенням буде обрати шаблон подієво-орієнтованої архітектури.

Подієво-орієнтована архітектура – це шаблон проектування програмного забезпечення який використовується для розробки графічних користувацьких інтерфейсів. Основна ідея подієво-орієнтованої архітектури полягає в тому, що програма розглядается як набір об’єктів, які реагують на певні події. Події можуть бути як зовнішніми або внутрішніми.

Зовнішні це такі події як натискання користувачем кнопки в програмі, зміна користувачем розміру вікна та ін. Внутрішні події це події які використовують об’єкти для комунікації один з одним. Такі об’єкти називаються об’єктами-обробниками.

Описати обробку подій можна наступним чином:

1) Програміст додає до системи графічного інтерфесу кнопку.
2) Створюється обробник подій який пов’язаний з цією кнопкою.
3) В залежності від встановленої програмістом задачі обробник подій виконує необхідні команди. Наприклад якщо кнопка при натисканні повинна змінювати колір заднього фону програми, то після натискання на кнопку обробник подій цієї кнопки передасть інформацію обробнику подій головного вікна програми про необхідність зміни кольору.

Для побудови програм з подієво-орієнтованою архітектурою середа розробки Visual Studio надає декілька API: Windows Forms та Windows Presentation Foundation (WPF).

Інтерфейс програмування додатків (API) – це опис способів якими одна комп'ютерна програма може взаємодіяти з іншою програмою. Одною з функцій

API є прийняття на себе реалізації частини функціональності програми, такої як графічний інтерфейс користувача.

Основна різниця між Windows Forms та WPF полягає в тому, що WPF використовує мову XAML для стильового оформлення елементів та відображення графіки без та приймає на себе задачі розробника з обробкою сцен анімації.

Так як розробляється ПЗ без необхідних функціональних вимог до відтворення складних графічних сцен та спецефічного стильового оформлення елементів, для проекту буде достатньо використання Windows Forms.

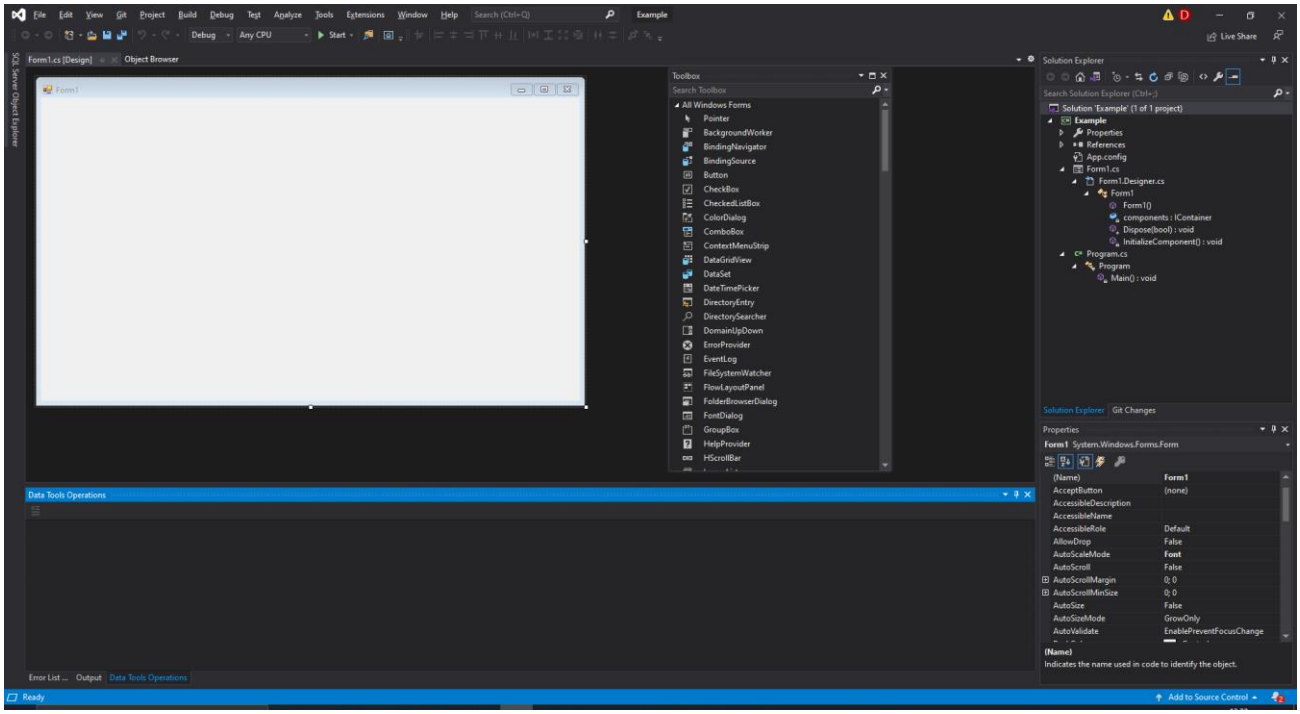


Рисунок 2.8 – API Windows Forms

При створенні нового проекту на Windows Forms автоматично створюються наступні файли проекту:

- Клас Program який є основною точкою входу для програми та запускає головне вікно.
- Головне вікно Form1 яке є контейнером для елементів керування та містить конструктор форми який визиває метод InitializeComponent();
- Клас Form1 який зберігає код та програмну логіку форми.

- Файл Form1.Designer є дизайнером форми, код якої середа Windows Forms створює автоматично та містить методи Dispose (), який виконує роль деструктора об'єкта, і InitializeComponent (), який встановлює початкові значення властивостей форми.
- Файл Form1.resx який зберігає ресурси форми.
- Файл App.config який відповідає за конфігурацію програми.

Панель Toolbox відображає елементи керування, які можна додати до проекту.

З панелі Toolbox можна перетягувати різні елементи керування на поверхню вікна, а також змінювати розмір та розташувати елементи керування.

Вікно Properties надає зручний інтерфейс для управління властивостями елемента та управління подіями елемента.

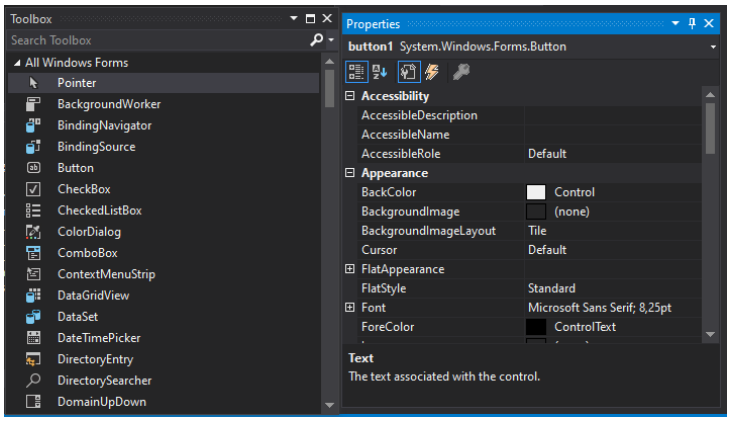


Рисунок 2.9 – Вікно Properties

При додаванні елементу керування на форму, та обиранні відповідної події, Windows Forms створює обробник подій яким є метод в класі Form1, з параметрами «object sender» який містить посилання на елемент управління та «EventArgs e» який містить дані про подію. (Рисунок 2.10).

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    this.BackColor = Color.Red;
}
```

Рисунок 2.10 – Створення обробника подій

2.3 Проектування системи

Опишемо систему на концептуальному рівні за допомогою діаграми прецедентів заснованої на функціональних вимогах для ПЗ, які було виведено раніше (Рисунок 2.11). Для створення діаграми буде використаний веб-сервіс Creately.

Creately – це веб-сервіс з схематичними інструментами, потужним контекстним інтерфейсом і спеціальними інструментами який використвується для створення діаграм. За допомогою цього сервісу можна створювати блок-схеми, UML-діаграми, мережеві діаграми, моделі бізнес-процесів, організаційні діаграми.

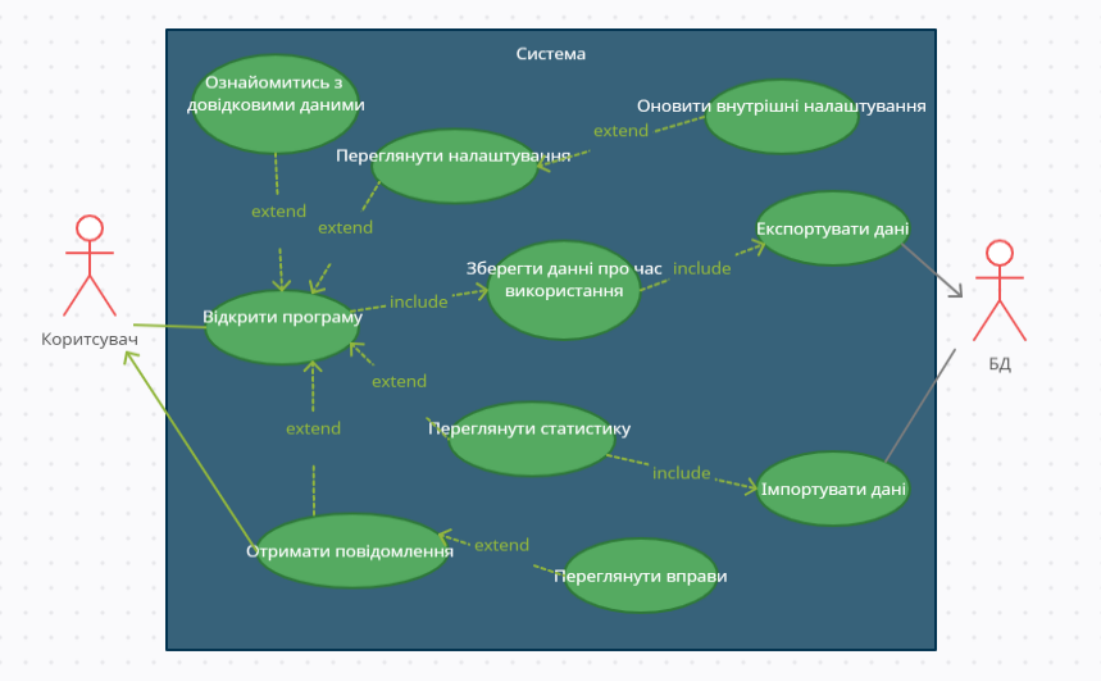


Рисунок 2.11 – Діаграма прецедентів

Опишемо алгоритм роботи системи:

При відкритті програми користувач може обрати, що саме він хоче зробити: ознайомитись з інформаційною сторінкою, перглянути налаштування та змінити їх при бажанні, або через деякий час отримати повідомлення про необхідність перерви чи повернення до роботи, у разі отримання повідомлення про перерву користувач також може переглянути вправи для зниження напруги з очей.

Дані налаштувань повинні зберігатись та завантажуватись безпосередньо з самої системи, оскільки користувач не повинен мати можливості змінювати їх напряму, тільки у панелі налаштувань самої програми.

Окремо слід відзначити імпорт та експорт статистичних даних про час проведений за роботою та відпочинком. Збереження даних має відбуватися регулярно і автоматично, а завантаження даних до системи тільки при запиті від користувача до їх перегляду.

Таке Відокремлення статистичних даних від системи потрібно для того щоб надати системі більшої гнучкості. Так у разі прийняття рішення про горизонтальне масштабування, завдяки відокремленості даних внесення змін буде набагато легшим.

Статистичні дані мають мають бути надаватись таким чином, щоб користувач одразу міг в них розібратись. Найкраще для цього підійде діаграма з відокремленими стовпцями.

Вікно з довідковими даними має містити в собі коротку інформацію яка надається користувачу в текстовому про КЗС та методи його усунення.

Знаючи необхідний функціонал програми, можна спроектувати графічний прототип головного вікна програми, який буде містити основні елементи керування. Надалі це допоможе в створенні користувацького інтерфейсу на етапі розробки, оскільки матиметься готовий шаблон. Для створення графічних прототипів буде використано Paint. Paint це графічний редактор компанії Microsoft, що входить до складу всіх операційних систем Windows та надається безкоштовно. Його функціоналу буде більш ніж достатньо для створення.

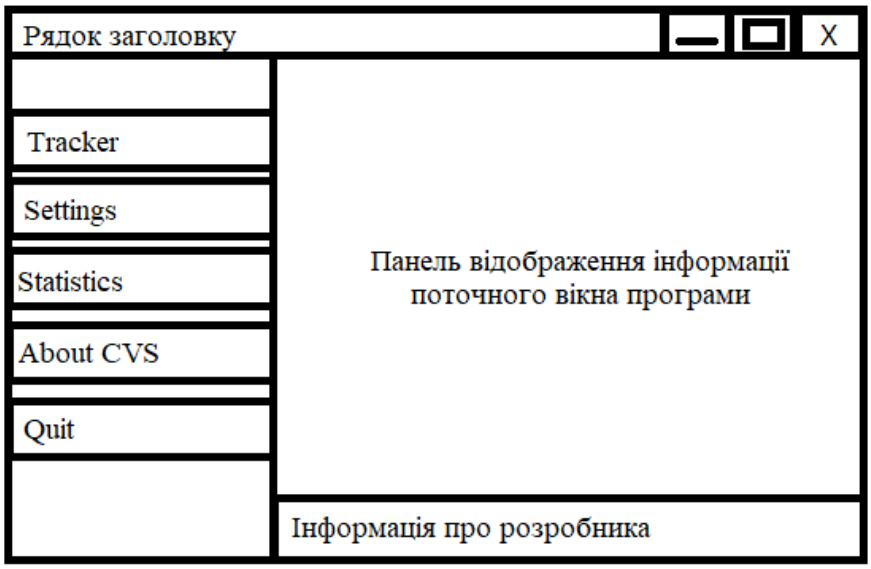


Рисунок 2.12 – Графічний прототипів меню

Кнопки Tracker, Settings, Statistics та About CVS при натисканні повинні виводити на панель відображення інформацію відповідну до їх назви.

Кнопка Quit відповідає лише за вихід з програми. Розглянемо принцип роботи кожної кнопки окремо.

При натисканні кнопки Tracker користувач повинен побачити таймер який буде відображати час до наступного відпочинку або повернення до роботи, в залежності від встановленого користувачем часу.

У тому випадку якщо час є часом для перерви, на панелі відображення повинні з’явитись вправи для очей які може переглядати користувач, та кнопка скасування яка відмінить поточний час відпочинку якщо користувач не бажає робити перерву в даний момент. Коли час відпочинку спливає або натиснута кнопка скасування, вправи зникають з панелі відображення інформації, а таймер оновлюється.

Рисунок 2.13 – Графічний прототип відображення вправ

Має сенс розробити алгоритм зчитування часу який буде змінювати дані які повинні відображатись на панелі. Для цього створимо діаграму діяльності яка буде відображати яким чином повинна змінюватись система.

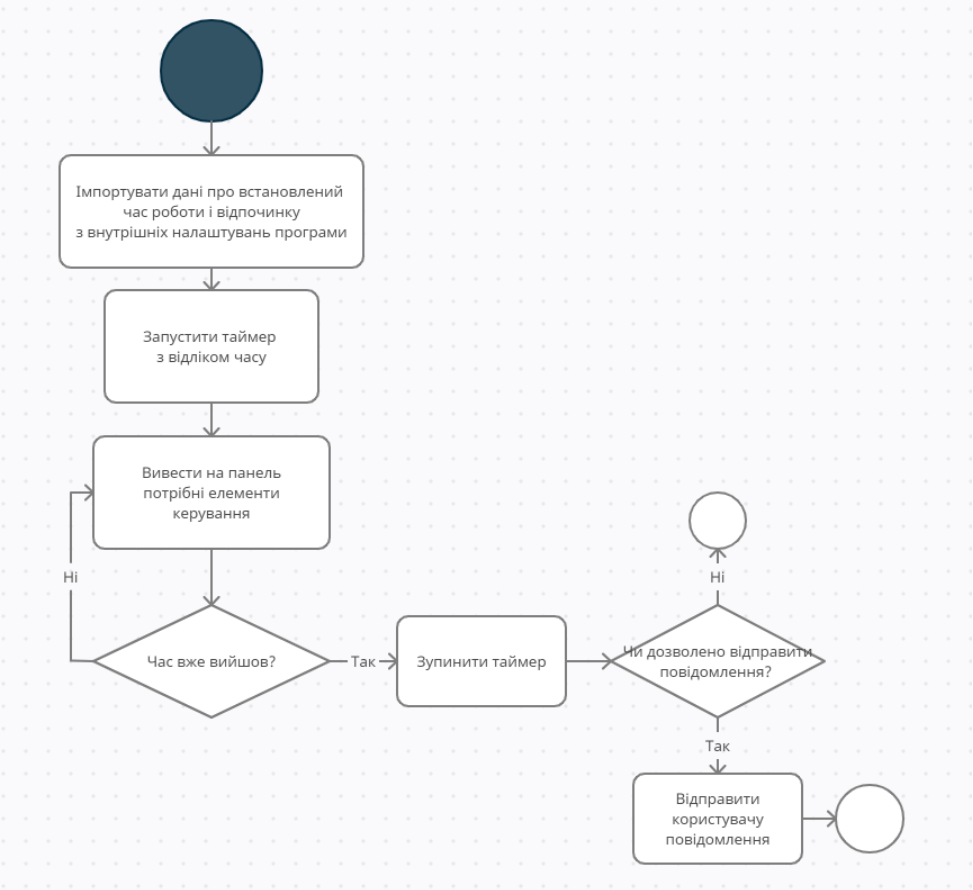


Рисунок 2.14 – Діаграма зміни системи

При запуску програми буде створівюватись таймер з кордоном відліку, який буде втановлюватись в залежності від налаштувань.

Поки таймер не досяг кордонного значення, на графічній панелі будуть відображатись відповідні дані. Якщо час вийшов дані перестають відображатись, а користувач в залежності від налаштувань отримає або не отримає повідомлення. Після цього таймер завеошить свою роботу.

При натисканні кнопки Settings на панель відображення виводиться вікно з можливістю налаштуваннями часу роботи та відпочинку, отримання повідомлень та відображення графічної складової програми, а саме можливість обрати світле або темне оформлення. Така функція дуже корисна для людей які працюють вночі, оскільки занадто яскраві кольори збільшують дискомфорт під час користування програмою.

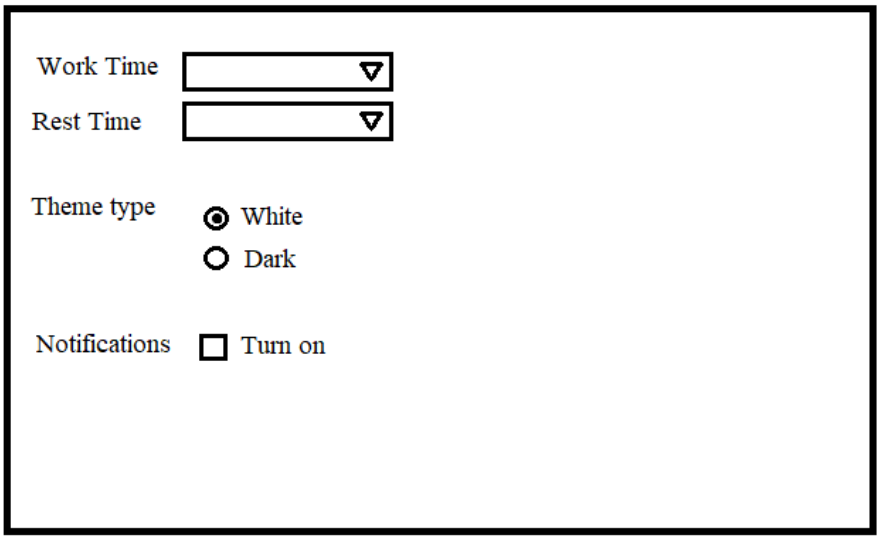


Рисунок 2.15 – Діаграма зміни системи

Збереження змінених налаштувань не потребує створення бази даних, оскільки Visual Studio передбачає можливість збереження даних налаштувань безпосередньо в самій програмі.

При натисканні кнопки About CVS на панель відображення виводиться текстова довідка про КЗС.

При натисканні кнопки Statistics на панель відображення виводиться діаграма з даними про час який користувач витратив на роботу та відпочинок протягом неділі. Оскільки виведений результат кнопок Statistics та About CVS будуть виводити лише по одному елементу, проектування їх графічного прототипу можна опустити.

Розумно буде надавати статистичні дані у певному часовому відрізку, так у користувача буде можливість проаналізувати набір даних та скоректувати час який він витрачає на роботу чи перерви. З цієї причини дані будуть зберігатись протягом неділі. Також системі потрібно знати за який саме період була зібрана ця статистика, для уникнення плутаниці. Приймаючи це до уваги, можна побудувати таблицю даних з яких повинна складатись БД.

Таблиця 2.2 Таблиця даних за останню неділю

Назва стовпця	Значення
Унікальний ідентифікатор дня	Int, унікальне, не дорівнює нулю
Назва дня	String, унікальне, не дорівнює нулю
Час роботи	int
Час відпочинку	int

Таблиця 2.3 Таблиця даних з неділею та роком

Назва стовпця	Значення
Унікальний ідентифікатор неділі	Int, унікальне, не дорівнює нулю
Номер неділі у році	Int
Номер року	Int

Оскільки обсяг статистичних даних незначний і уміщуються у дві таблиці, було б марнотратно створювати повноцінну базу даних. Замість цього можна скористатися технологією ADO.NET автономного рівня.

ADO.NET – це технологія, що надає доступ і керування даними, що зберігаються в базі даних. Її автономний рівень надає можливість замість створення клієнт-серверних систем, створювати об'єкти DataSet, які націлені на автономну роботу. Це дозволяє створювати таблиці прямо у програмі, не підключаючись до бази даних. При цьому ADO.NET надає увесь необхідний для їх конструювання автономних типів, а у разі вирішення про підключення програми до більшої бази даних, може надати адаптер даних для автоматичного обслуговування підключення [8].

Через те що статистичні дані не можна віднести до внутрішніх налаштувань програми, для експорту та імпорту даних потрібно буде використовувати базу даних та зберігати її в окремих файлах. Тому перш ніж приступати до побудови програмного забезпечення, потрібно визначити як саме буде проходити взаємодія між базою даних та системою.

Коли користувач натискає кнопку Statistics, проходить перевірка на існування файлів БД у відповідній папці. Якщо файлів не знайдено, що може трапитсь коли користувач вперше запустив програму, система створює необхідні файли сама. Після цього система зчитує дані з БД. У тому випадку якщо дата БД та дата дня коли користувач запустив програму повністю збігаються, проводиться перевірка на наявність даних за цей день у БД. Це потрібно через те що користувач може мати декілька робочих сесій на день, між якими він вимикав комп’ютер. Так можна зберегти дані не тільки поточної сесії, а сесії за весь день. Якщо ж дати БД і користувача не збігаються, проводиться перевірка на різницю між датами.

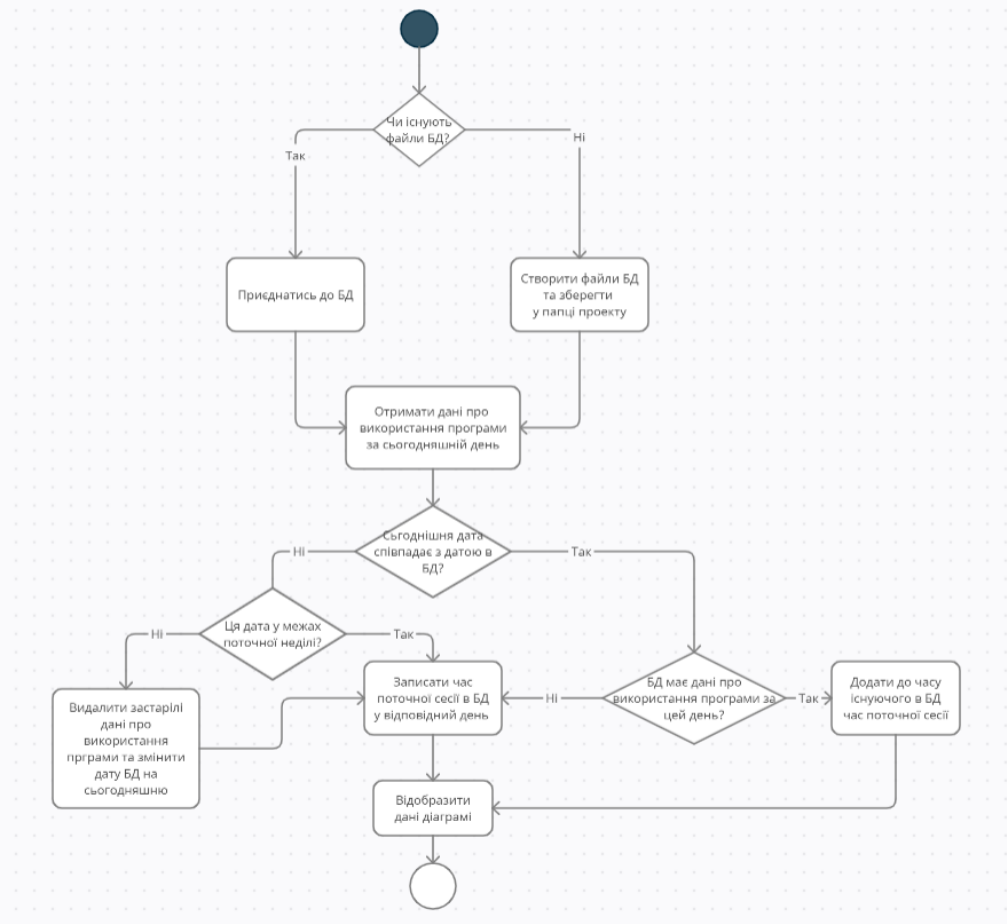


Рисунок 2.16 – Діаграма взаємодії системи та БД

У разі змінення лише дня неділі, БД просто оновить свою дату та запише дані отримані від користувача до відповідного розділу. Якщо дата користувача перевищує значення поточної неділі (наприклад якщо користувач запустив програму через рік після останнього використання) БД очищує усі дані робочих сесій, та встановлює дату системи користувача.

Висновок до розділу

У розділі була обрана операційна система, середа розробки, необхідні для розробки технології та мова програмування. Створені прототипи та діаграми на базі функціональних вимог проекту.

3 ПОБУДОВА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Реалізація користувацького інтерфейсу

Маючи необхідні графічні прототипи користувацького інтерфесу, можна перейти до його безпосередньої реалізації у API Windows Forms.

Спершу необхідно реалізувати головне вікно програми, оскільки саме через нього буде надаватись основна інформація. (Рисунок 3.1).

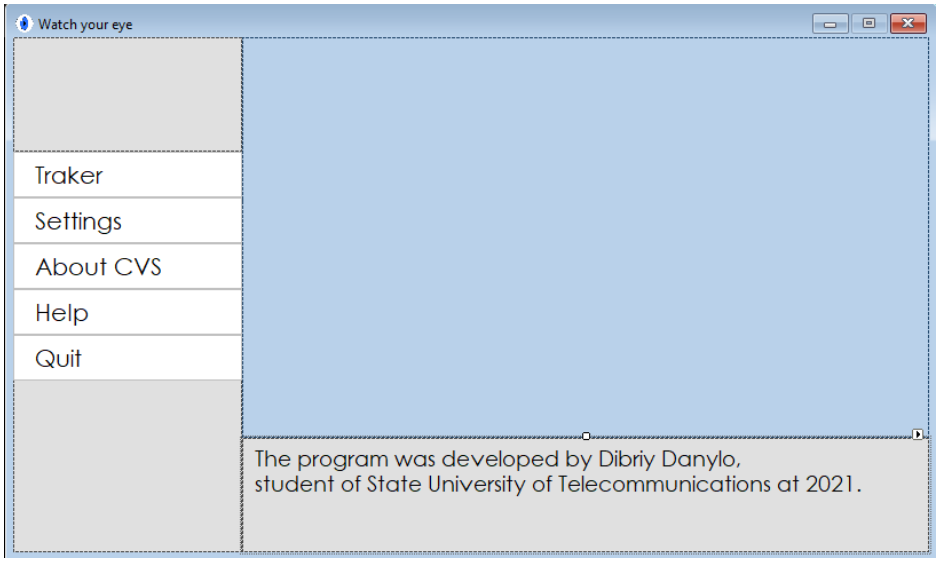


Рисунок 3.1 – Головне вікно програми

Для створення кнопок був використаний елемент керування Button.

У якості панелі для відображення даних послугував елемент керування Panel, він служить для групування елементів управління та допомагає розділити вікно за функціями.

При натисненні одної з кнопок виведення інформації, у панелі буде створюватись необхідне вікно. Вікна повинні будти відокремлені, така модульність значно облегшить роботу у тому разі якщо буде необхідно масштабувати програму у майбутньому. Зважаючи на це, були створені окремі вікна для кожного кнопки яка виводить дані.

Для реалізації вікна яке виводить для користувача вправи та демонструє час було використано елементи PictureBox, задача якого полягає в виведенні графічних зображень. Саме через нього будуть демонструватись вправи.
Елементи Button зі стрілками слугують для руху по колекції вправ.
Кнопка Stop слугує для скасування відпочинку.

Інформація про спилинувший час відображається за допомогою компоненту Label, який використовується для відображення невеликих за розмірами текстів який не може бути змінений користувачем.

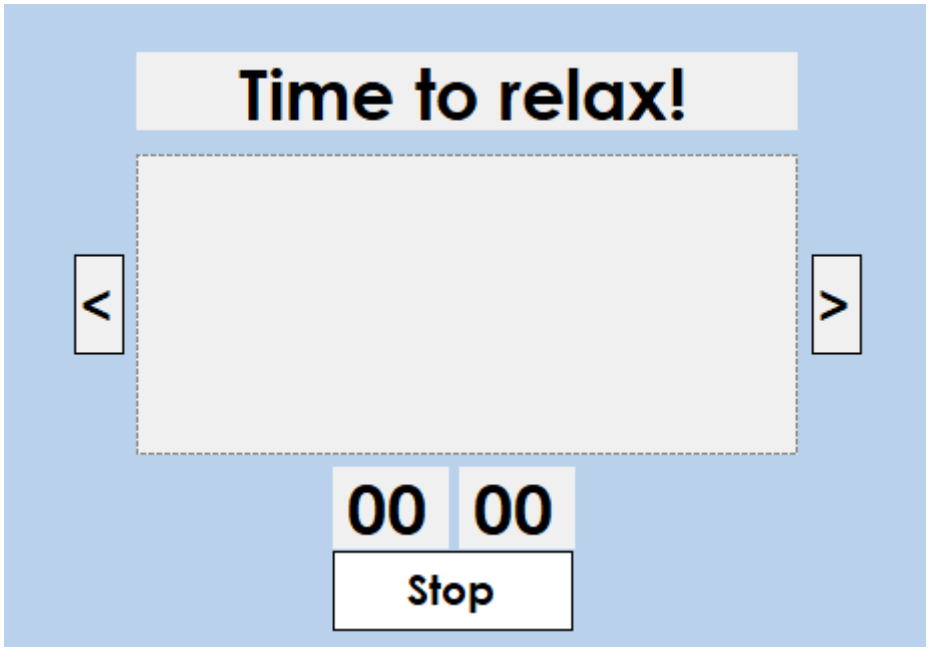


Рисунок 3.2 – Реалізація інтерфесу відображення вправ

Для створення вікна налаштувань був використаний елемент ComboBox, який використовується для зберігання елементів у випадаючому списку, у списках будуть зберігатись варіанти для налаштування часу.

Також були використані елементи RadioBatton та CheckBox, вони використовуються для встановлення одного з двох можливих значень, еквівалент true/false.

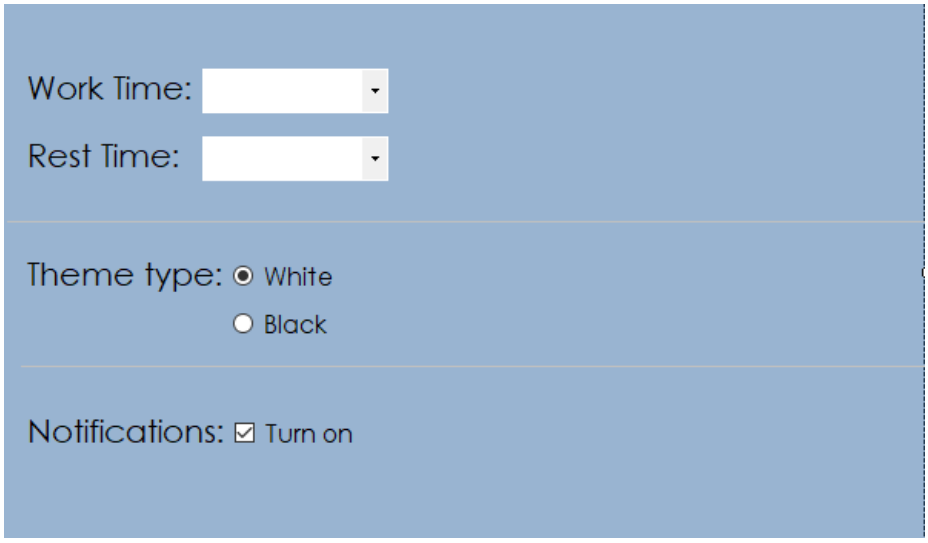


Рисунок 3.3 – Реалізація інтерфесу панелі налаштувань

Для вікна статистики був використаний лише елемент Chart, який відповідає за відображення даних. Для відображення часу роботи та часу відпочинку було модифікована властивість Series. На даному етапі відображені ніякі дані не відображаються, доки елемент не отримає даних з БД. Дані продемонстровані зараз є лише можливим прикладом. (Рисунок 3.4).

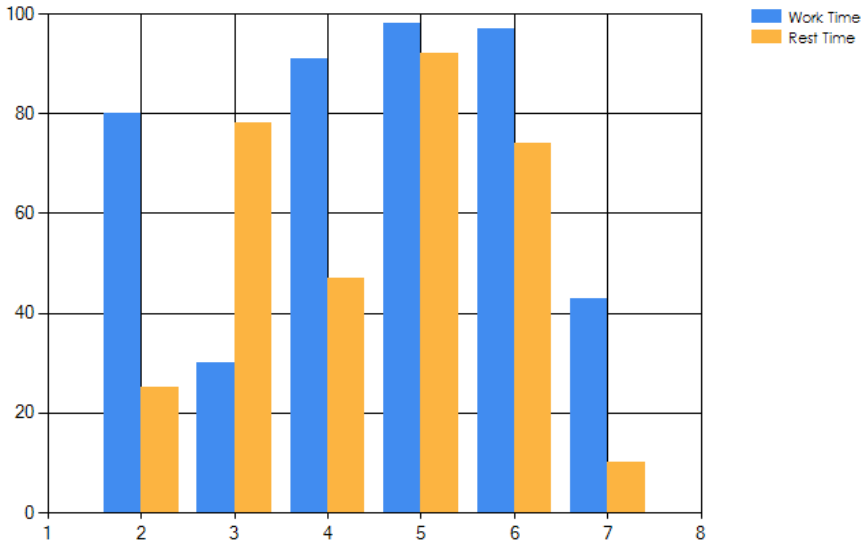


Рисунок 3.4 – Реалізація вікна статистики

Для реалізації вікна About CVS був використаний елемент RichTextBox, який слугує для відображення, введення та обробки тексту з форматуванням. (Рисунок 3.5).

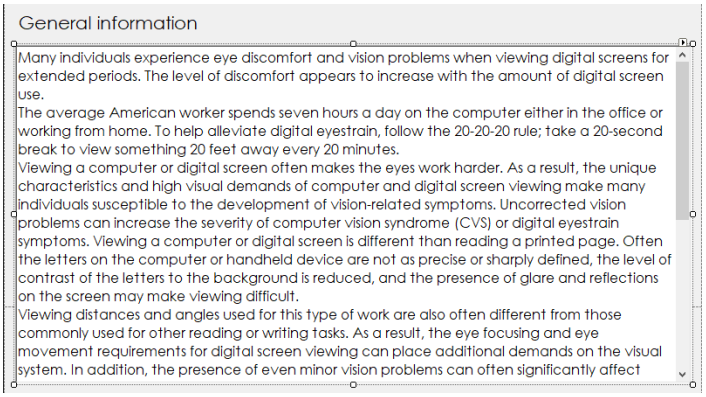


Рисунок 3.5 – Реалізація вікна довідки про КЗС

Для викликання вікон при взаємодії з відповідними кнопками створимо у класі головного вікна створимо метод Open_child_form() , який у якості параметра приймає вікно яке необхідно відобразити в панелі при натисканні кнопки.

```
private void Open_child_form(Form child_form)
{
    if (active_form != null)
    {
        active_form.Close();
    }
    active_form = child_form;
    child_form.TopLevel = false;
    child_form.FormBorderStyle = FormBorderStyle.None;
    child_form.Dock = DockStyle.Fill;
    main_panel.Controls.Add(child_form);
    child_form.BringToFront();
    child_form.Show();

}
```

Рисунок 3.6 – Взаємодія вікон

При його виклику попередньо відкрите вікно буде закриватись, а нове повністю заповняти простір наданий панелю.

3.2. Реалізація алгоритмів зчитування часу та взаємодії з БД

Для реалізації алгоритму який буде зчитувати час спершу необхідно додати до головної панелі компонент Timer.

Компонент Timer слугує для викликання подій через рівні проміжки часу. Оскільки буде відраховуватись реальний час, властивість таймера під назвою Interval треба встановити на 1000 (властивість приймає значення в мілісекундах).

Через те що час потрібно зчитувати відразу після запуску програми, його запуск повинен здійснюватись в класі головної форми програми.

При старті починається подія таймеру яка виражена у методі timer1_Tick.

Якщо це таймер для відліку часу роботи, він передає у метод Time_counter елементи Label, максимально допустиме значення часу та відображає відповідні графічні елементи доки поки метод Time_counter не поверне false.

Після того як час таймеру сплинув, користувач може откримати повідомлення в залежності від того чи настав зараз час відпочинку або повернення до роботи. Це виконується за допомогою визову методу ShowNotifications(), який приймає у якості параметру строку даних яка буде виведена в повідомлені.

Якщо користувач увімкнув відображення повідомлень у налаштуваннях воно буде відображене у нижньому правому куті монітору.

```
1 reference
private void timer1_Tick(object sender, EventArgs e)
{
    Switch_theme();
    if (time.Time_counter(seconds_label, minutes_label, Settings.Default.Work_time) == true)
    {
        timer1.Enabled = false;
        timer2.Enabled = true;
        Stop_rest_button.Visible = true; Stop_rest_button.SendToBack();
        switch_left_button.Visible = true; switch_left_button.SendToBack();
        switch_right_button.Visible = true; switch_right_button.SendToBack();
        pictureBox1.Visible = true; pictureBox1.SendToBack();
        ShowNotifications("Time to rest!", "Take your eyes off the screen and let them rest");
    }

}

1 reference
private void timer2_Tick(object sender, EventArgs e)
{
    Switch_theme();
    relax_label.Visible = true;
    if (time2.Time_counter(pause_s, pause_m, Settings.Default.Rest_time, Settings.Default.is_seconds) == true)
    {
        relax_label.Visible = false;
        timer2.Enabled = false;
        timer1.Enabled = true;
        Stop_rest_button.Visible = false;
        switch_left_button.Visible = false;
        switch_right_button.Visible = false;
        pictureBox1.Visible = false;
        ShowNotifications("Back to work!", "You can keep working");
    }
}
```

Рисунок 3.7 – Реалізація алгоритму роботи таймеру

```
2 references
void ShowNotifications(string heading,string text)
{
    if (Settings.Default.Show_notifications)
    {
        notifyIcon.Icon = SystemIcons.Exclamation;
        notifyIcon.ShowBalloonTip(1000, heading, text, ToolTipIcon.Info);
    }
}
```

Рисунок 3.8 – Реалізація відображення повыдомлень

Запустимо програму для перевірки щоб переконатися що все працює коректно та отримаємо наступний результат:

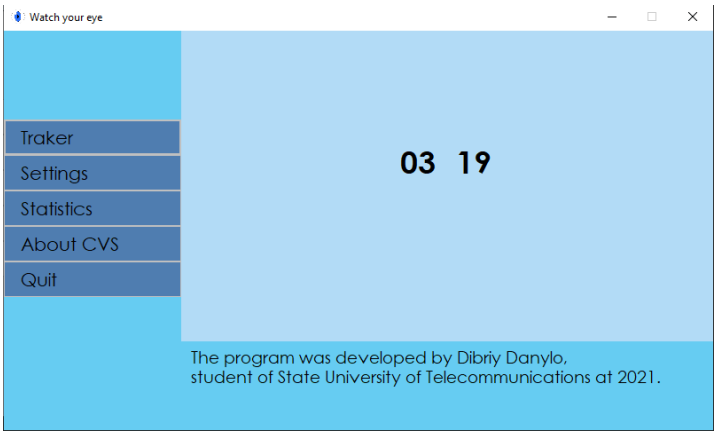


Рисунок 3.9 – Відображення спилнувшого часу роботи

Пілся сплинення часу на панелі з’явились нові елементи та було отримано повідобення про початок відпочинку.

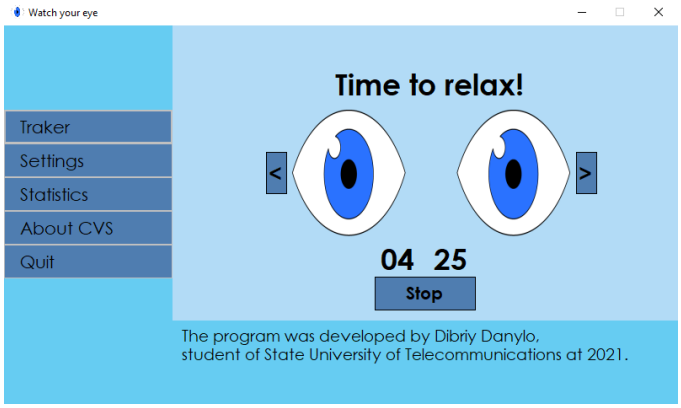


Рисунок 3.10 – Відображення вправ під час відпочинку

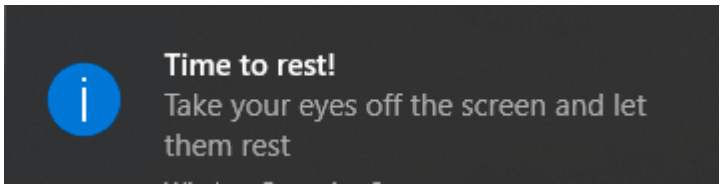


Рисунок 3.11 – Отримане повідомлення

Як бачимо, таймер коректно виконує свої функції.

Оскільки було вирішено використовувати технологію ADO.NET автономного рівня для побудови бази даних, можна створити методи для зберігання, оновлення та побудови таблиць прямо у програмі.

Через те що дані необхідні лише для демонстрації діаграми у вікні статистистики, виклик відповідних методів має відбуватися у разі відкриття відповідного вікна. Для підвищення модульності архітектури програми, ці методи були винесені до окремого класу з ім’ям Statistic_data який підтримує методи Build_Data() для побудови таблиць, Write_data() для запису даних, Save_data() для їх збереження у папці ‘saved_data’ яка знаходиться у каталозы в якому програма розміщена, та Load_data() для їх завантаження.

За допомогою методу FillData() у класі вікна статистики, дані про час роботи та відпочинку імпортуються до таблиць, і у разі їх коректності зберігаються в них. Після цього за допомогою об’єкту DataTable програма запрошує дані з таблиці де зберігається інформація та відображає її на діаграмі. (Рисунок 3.11).

```
1 reference
void FillData()
{
    Statistic_data data = new Statistic_data();
    data.WorkTime = worktime;
    data.RestTime = resttime;
    data.Write_data(current_day);


    DataTable chart_table = data.statistic_data.Tables[0];

    foreach (DataRow row in chart_table.Rows)
    {
        work_chart.Series["Work Time"].Points.AddXY(row["DayName"], (int)row["WorkTime"]);
        work_chart.Series["Rest Time"].Points.AddY((int)row["RestTime"]);
    }

}
```

Рисунок 3.12 – Метод для відображення інформації з БД

Перевіримо роботу алгоритму експорту та імпорту. Для цього запустимо програму і почекаємо деякий час, після цього при натисканні на кнопку ‘Statistics’ ми повинны графік з інформацією про те скільки часу ми витратили.

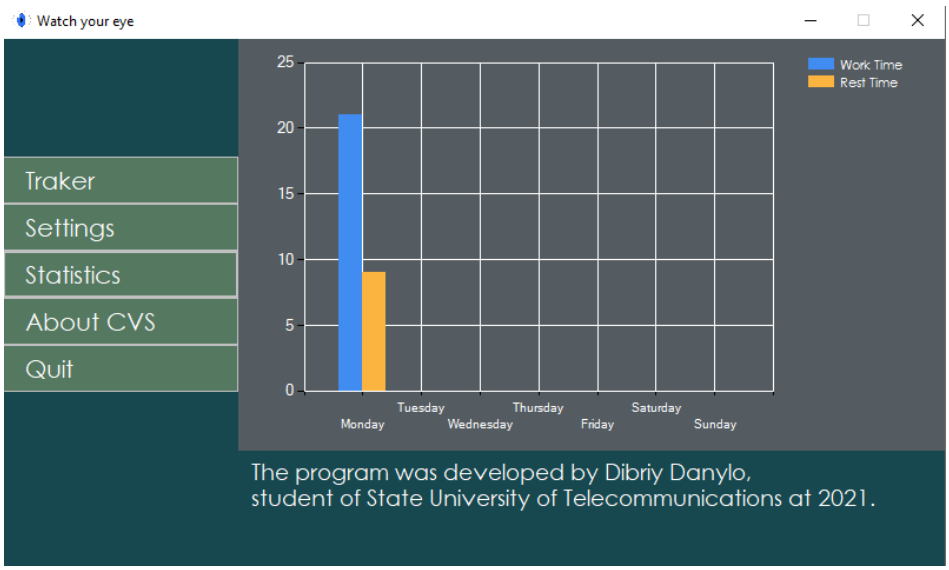


Рисунок 3.13 – Отримана статистика

Перевіримо як поведе себе програма при зміні дня неделі на середу. У разі правильної реалізації алгоритму, дані почнуть записуватись до відповідного розділу. (Рисунок 3.14).

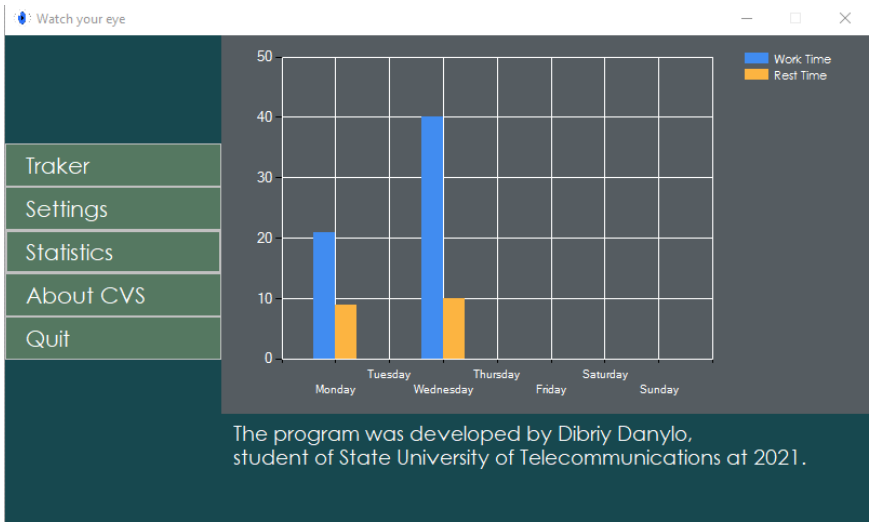


Рисунок 3.14 – Отримана статистика

Як бачимо, місце запису даних змінилося правильно. Останнє що необхідно перевірити це очищення даних якщо дата системи користувача перевищує дату записану в БД далі поточного тижня. Для встановимо дату системи на наступний понеділок. (Рисунок 3.15).

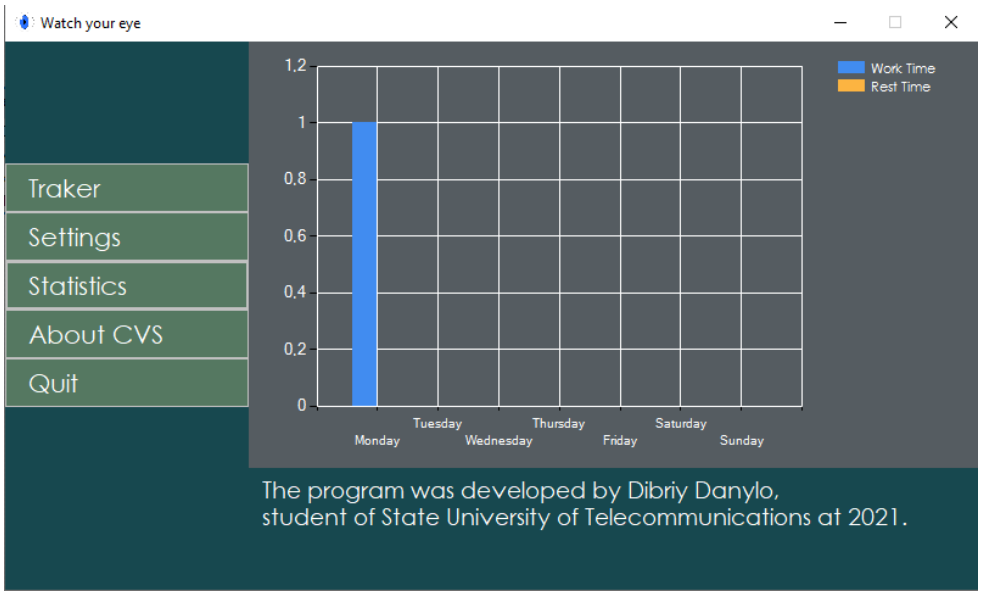


Рисунок 3.15 – Отримана статистика після початку нового тижня

Як можна побачити, після зміни дати інформація про попередній тиждень була повністю видалена та почала записуватись заново.

Після проведення ряду тестів можна зробити висновок алгоритм працює коректно.

3.3 Реалізація збереження налаштувань користувача

Для збереження внутрішніх налаштувань додамо до програми файл Settings. Settings дозволяє легко створювати, зберігати та підтримувати власні налаштування програми та налаштування користувача.

Додамо до файлу наступні значення:

- Work_time типу Int, який буде зберігати дані встановленого користувачем часу роботи.
- Rest_time типу Int, який буде зберігати дані встановленого користувачем часу перерв.
- White_theme типу bool, для зберігання даних про зовнішній вигляд інтерфейсу користувача.
- is_seconds типу bool, для зберігання інформації про те які перерви хоче робити користувач – секундні чи хвилинні.
- Show_notifications типу bool для зберігання даних про те чи хоче користувач бачити отримувати повідомлення від програми.

Name	Type	Scope	Value
Work_time	int	User	20
Rest_time	int	User	20
White_theme	bool	User	True
is_seconds	bool	User	True
Show_notificati...	bool	User	True

Рисунок 3.16 – Зміст файлу Settings

Для перевірки коректної роботи спробуємо змінити тип відображення інтерфейсу на темний та закрити програму.

У разі коректної роботи файлу Settings при новому відкриті програми він буде відображатись у темній темі.

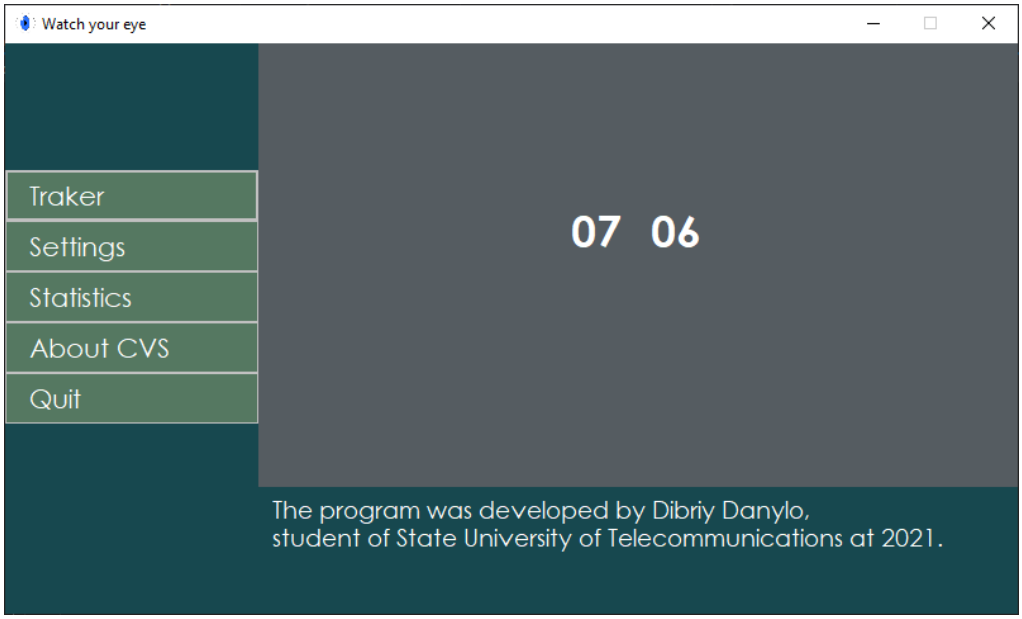


Рисунок 3.17 – Відображення інтерфейсу після зміни теми

Після при нового відкриття програми вона відобразилась елементи інтерфейсу змінили колір на більш темний, тому можна зробити висновок про коректну роботу файлу Settings.

3.4 Створення інсталятору та тестування програми

Для можливості комфортного розповсюдження програми на інші ПК необхідно створити інсталятор, який зможе встановики програму у вибрану користувачем папку.

Для цього до проекту який містить програму потрібно додати ще один, потрібно додати ще один проект типу Setup Project, який прийме на себе обов'язки по створенню інсталятора. (Рисунок 3.18).

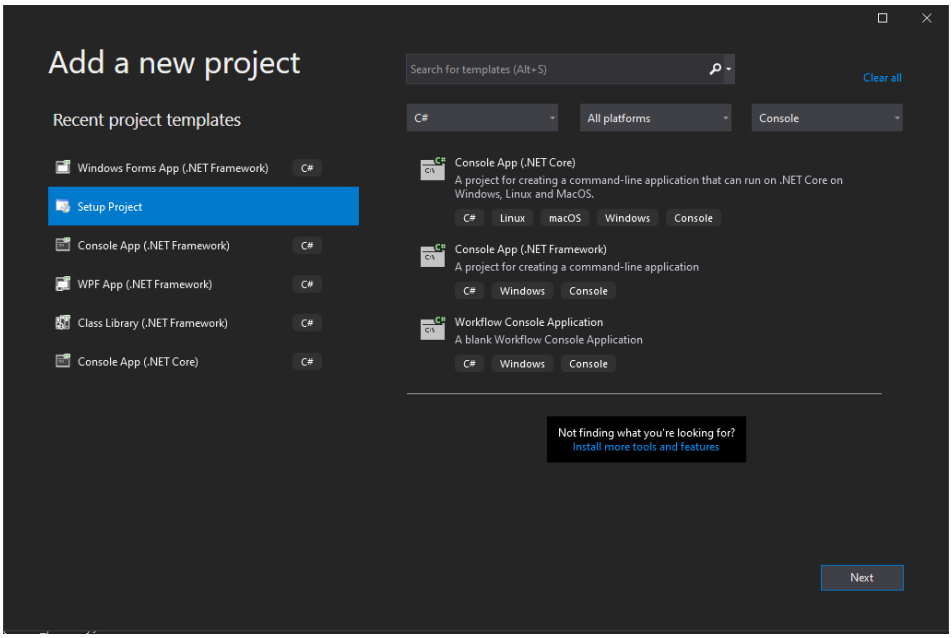


Рисунок 3.18 – Додавання проекту Settings

Після цього необхідно обрати яка сама частина програми повинна міститись у інсталяційному файлі, чи повинні створюватись додаткові папки, як саме будуть відображатись ярлики програми та ін..Оскільки нам необхідні тільки файли запуск програми , обираемо Primary output. (Рисунок 3.19).

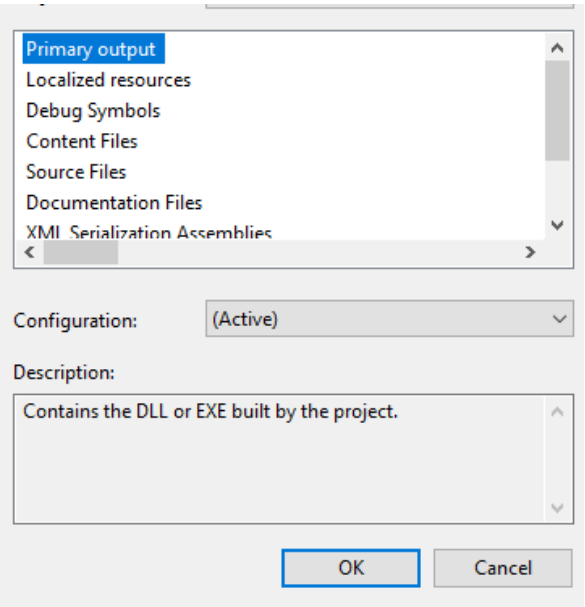


Рисунок 3.19 – Додавання проекту до інсталятора

У вікні властивостей інсталятора потрібно видалити стандартні значення та встановити власні, так можна буде дізнатись про те хто розробив програму. Встановимо опис, автора та ім’я розробника, та змінимо текст заголовок інсталятора.

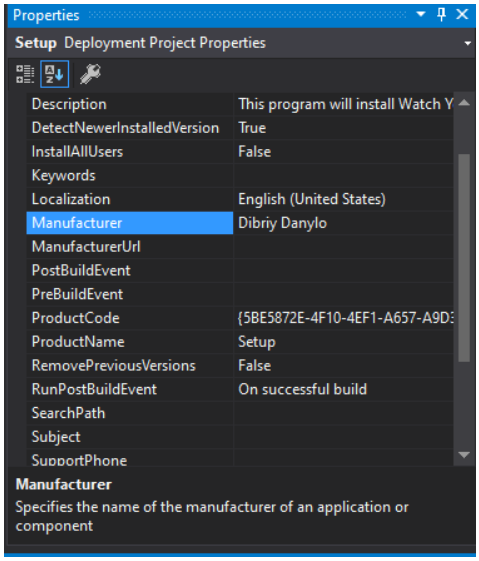


Рисунок 3.20 – Налаштування властивостей проекту Settings

Після встановлення усіх необхідних налаштувань інсталятора та додавання проекту потрібно лише запустити його. Якщо на етапі збірки не виникне помилок, з'явится вікно інсталялтора. (Рисунок 3.21).

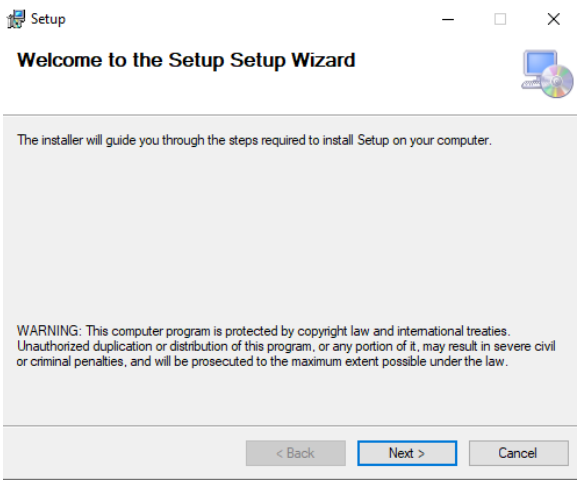


Рисунок 3.21 – Вікно інсталяції

Після побудови програми потрібно переконатись у відсутності проблем, для цього програма повинна пройти тестування.

Тестування програмного забезпечення - процес аналізу програмного засобу та супутньої документації з метою виявлення дефектів і підвищення якості продукту.[9] Тесування проводится як для переконання у коректної роботи програми відповідно до функціональних вимог та не функціональних вимог, так і для доказу непрацездатності програми в деяких заданих умовах.

Тестування може бути ручним та автоматичним. При ручному тестуванні тестувальник не використовує допоміжних програмних засобів, а при автоматичному – навпаки.

Оскільки програма достатньо мала у своєму обсязі, буде викоритсано ручне тестування. Існує досить велика кількість підходів до тестування для знаходження помилок в програмі, для тестування програми було обрано наступні:

- Димове тестування – націлено на перевірку ключової функціональності, непрацездатність якої робить безглуздою використання програми.
- Тестування критичного шляху – націлено на дослідження функціональності, використовуваної типовими користувачами в типовій повсякденній діяльності.
- Позитивне тестування – націлено на дослідження програми в ситуації, коли всі дії виконуються строго по інструкції без яких би то не було помилок, відхилень, введення невірних даних.
- Негативне тестування – націлено на дослідження роботи програми в ситуаціях, коли з ним виконуються некоректні операціїі які потенційно призводять до помилок.

При знаходженні помилки необхідно систематизувати інформацію про неї, для цього використовуються звіти про помилки. Для створення та систематизації звітів про помилки букло використано сервіс Jira.

Jira – це система для відстеження помилок та роботи з проектами. Платформа надає гнучкий інструментарій для зміни зовнішнього вигляду інтерфейсу для поліпшення роботи користувача.

При створенні звіту про помилку в Jira треба запованити наступні поля:

- Проект – тут вказується поточний проект.
- Тип задачі – вказується тип звіту.
- Резюме – містить короткий опис проблеми.
- Опис – містить детальний опис проблеми.
- Кроки відтворення – як саме можна відтворити помилку.
- Пріорітет – наскільки помилка критична.
- Вкладення – додаткова інформація про помилку, така як скріншоти.

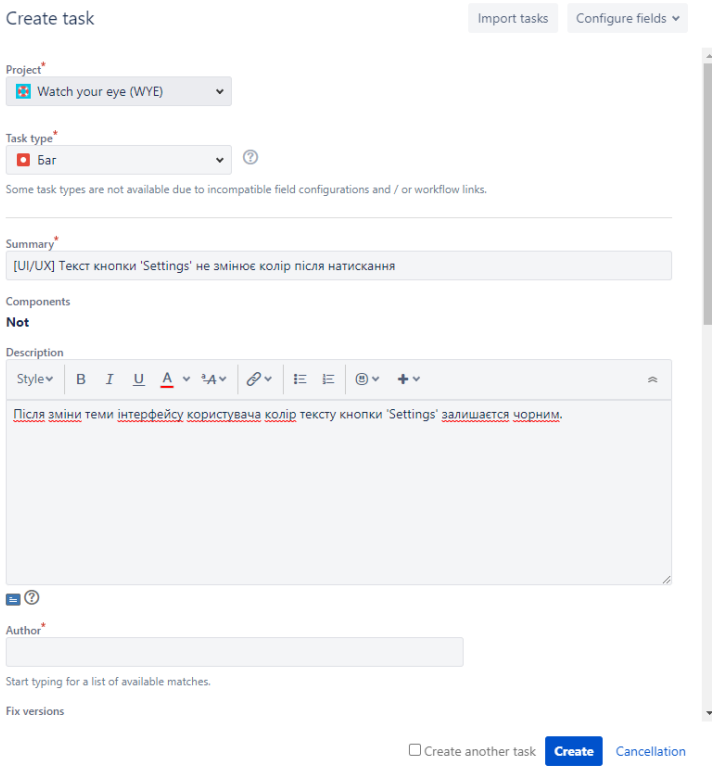


Рисунок 3.22 – Створення звіту про помилку

Для планування та перевірки стану тестування потрібно розробити набір тест-кейсів для проекту.

Тест-кейс – це документація тестувальника, що описує як прийти до фактичного результату.

Для створення тест-кейсів був використаний сервіс TestRail.

TestRail це сервіс який допомагає та відстежувати стан тестування програмного забезпечення. Його інтуїтивно зрозумілий веб-інтерфейс дозволяє легко створювати тестові кейси, керувати тестовими запусками та координувати весь процес тестування.

Для створення тест-кейсу у TestRail необхідно заповнити наступні поля:

- Title – заголовок який описує тест-кейс.
- Preconditions – передумови які необхідно виконати пере основними кроками.
- Steps – кроки виконання.
- Type – вид тест кейсу.
- Priority – наскільки помилка тест-кейс важливий.
- Expected result – очікуваний результат після виконання.

Після виконання тест кейсу потрібно заповнити дані про результат його проходження. Якщо після проходження тест кейсу не було знайдемо помилок, його можна відмітити як “Passed”. Якщо після проходження тест кейсу були знайдені помилки які повністю не перешкоджають роботі, то ставиться “Retest”. У тому разі коли помилка повністю перешкоджає роботі ставиться ‘Failed’.

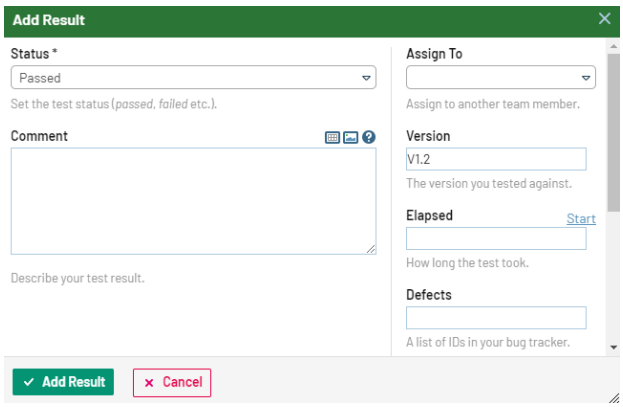


Рисунок 3.23 – Відкно заповнення результатів тестування тест кейсу

Якщо всі тест кейс були закриті як «Passed», тестування поточної версії завершується. Якщо ж це фінальна версія продукту, значить тестування можна вважати повністю закінченим. (Рисунок 3.24).

Рисунок 3.24 – Завершення тестування

Висновок до розділу

У розділі був описаний процес побудови програмного забезпечення для зберігання здоров'я очей під назвою «Watch Your Eye». Також було проведене міжетапне тестування коректності роботи алгоритмів створених у попередньому розділі. Після створення інсталятора програми було проведене повне тестування продукту за допомогою різних методів тестування та закриті усі тест кейси, що означає що побудова проекту завершена.

ВИСНОВКИ

У даній роботі були досліджені методи збереження здоров’я очей під час роботи за комп’ютером та створено відповідне ПЗ десктопне ПЗ на мові мові C#. Створена прикладна програма має велику соціально-практичну значущість, оскільки допомагає людям які працюють за екранними пристоями та звичайним користувачам зберегти свій зір та покращити загальне самопочуття.

У першому розділі дипломної роботи було проведене дослідження наукових журналів та наказами Міністерства охорони здоров'я України, та були визначені симптоми КЗС та необхідні профілактичні практики для збереження зору. Також були проаналізовані існуючі системи по збереженню зору та виявлено їх недоліки що не дають повністю проводити профілактику КЗС.

У другому розділі були проаналізовані методології розробки ПЗ та за допомогою порівняння їх характеристик обрвна гнучка методологія розробки, як найбільш підходяща до проекту. Надалі були створені вимоги продукту та обрано операційну систему Windows, середу розробки Microsoft Visual Studio та інтерфейс програмування додатків Windows Forms за допомогою яких можна було створити продукт. Були створені прототипи інтерфейсу програми, алгоритм взаємодії БД та сисеми, алгоритм роботи таймеру та відправлення повідомлень.

У третьому розділі було описано побудову ПЗ, створенений інсталятор продукту, та проведено тестування за допомогою сервісів Jira та TestRail.

У результаті було виконано усі завдання визначені у вступі та створено робоче десктопне програмне забезпечення для зберігання здоров’я очей під час роботи за комп’ютером на мові C#.

Основні положення і результати бакалаврської роботи доповідались і обговорювались на науково-практичніх конференціях:

1. Дібрій Д.А. Розробка ПЗ для автоматизації процесу зберігання здоров’я очей користувача під час роботи за ПК / Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інфокомунікаційних технологія».

2. Дібрій Д.А. Розробка десктопного ПЗ для зберігання здоров’я очей користувача на мові програмування C# / Всеукраїнська науково-технічна конференція «Сучасний стан та перспективи розвитку IOT».
3. Дібрій Д.А. Актуальність створення ПЗ для зберігання здоров’я очей користувача / Всеукраїнська науково-технічна конференція «Сучасний стан та перспективи розвитку IOT».

Перелік посилань

1. Akinbinu T.R. , Mashalla Y.J. Imact of computer technology on health: Computer Vision Syndrome(CVS)
2. Hayati Joshi, Seemi Retharekar The effect of eye exercises on visual acuity and refractive error of myopics.
3. World Health Organization, World report on vision
4. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98.
5. Charles Boulet, The '20/20/20 Rule' – When Good Intentions and Axiomatic Habit Displace Best Practices
6. Mac App Store Breaktime Ratings and reviews [Електронний ресурс] – Режим доступу: https://apps.apple.com/ru/app/breaktime/id427475982?mt=12#see-all/reviews
7. Desktop Operating System Market Share Worldwide, StatCounter Global Stats [Електронний ресурс] – Режим доступу: https://gs.statcounter.com/os-market-share/desktop/worldwide
8. Troelsen Andrew, pro c# 5: with .net and .net core
9. Куліков С.С. Тестування програмного забезпечення.
10. Домашня сторінка документації та навчальних ресурсів Майкрософт для розробників і технічних фахівців [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/ru-ru/
11. Creately [Електронний ресурс] – Режим доступу: https://app.creately.com/
12. Jira [Електронний ресурс] – Режим доступу: https://www.atlassian.com/ru/software/jira
13. Testrail [Електронний ресурс] – Режим доступу: https://www.gurock.com/testrail/
14. Фраулер М. UML Основи.

Додаток

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка десктопного програмного забезпечення для зберігання здоров'я очей під час роботи за комп'ютером на мові c#

Виконав студент 4 курсу
групи ПД-41
Дібрій Данило Андрійович
Керівник роботи
Трінтіна Наталія Альбертівна

Київ – 2020

Актуальність теми

Через стрімкий розвиток інформаційних технологій все більше людей починає проводити більше значний час за екранами комп'ютерів, що призводить до появи професійних захворювань. Одним з цих захворювань є комп'ютерний зоровий синдром(КЗС), симптоми якого ведуть до погіршення зору. Згідно останніх досліджень відсоток студентів які мають проблеми з зором інженерних спеціальностей становить 81.9%.

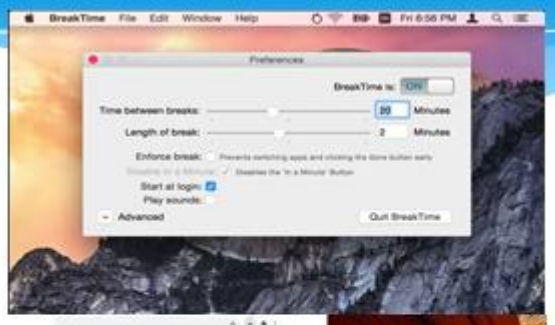
МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Метою роботи** є створення десктопного користувацького ПЗ для зберігання здоров'я очей користувача під час роботи за комп'ютером на мові C#.
- **Об'єктом дослідження** є методи зберігання здоров'я очей під час роботи за комп'ютером.
- **Предмет дослідження** є десктопне користувацьке програмне забезпечення для зберігання здоров'я очей користувача відповідно до вимог українського законодавства та сучасних офтальмологічних досліджень.

Аналоги

Вже існує невелика кількіст
програм яка дозовляє прове
профілактику КЗС, але у ко
з них існує ряд недоліків як
створюють певні незручнос

Розглянемо загальні позити
негативні сторони існуючог



Аналіз аналогів

Позитивні сторони

- Має можливість налаштування часу перерв
- Прості в використанні
- Не займають багато місця на комп'ютері

Негативні сторони

- Надає недостатню кількість варіантів для зміни часу
- Не все ПЗ надає вправи для очей та можливості налаштування повідомлень
- Жодна з програм не надає користувачу статистики використання

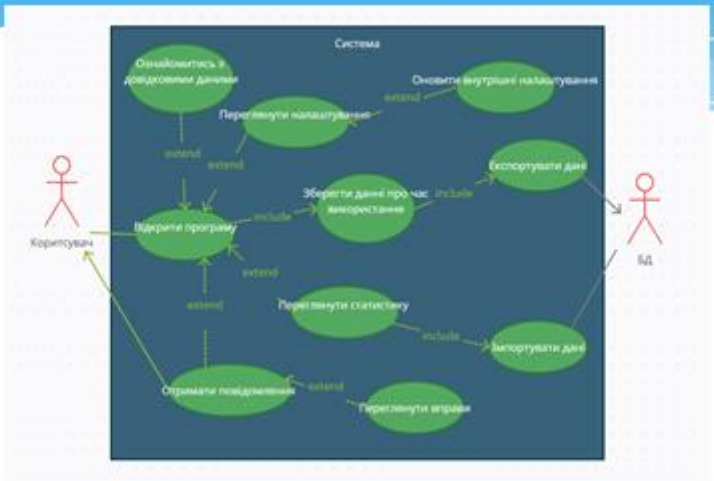
Технічне завдання

- Проектування системи.
- Реалізація методів для профілактики КЗС у ПЗ з графічним інтерфейсом.
- Реалізація функцій призначених для користувача налагоджень.
- Реалізація функції отримання статистичних даних користувача з БД.

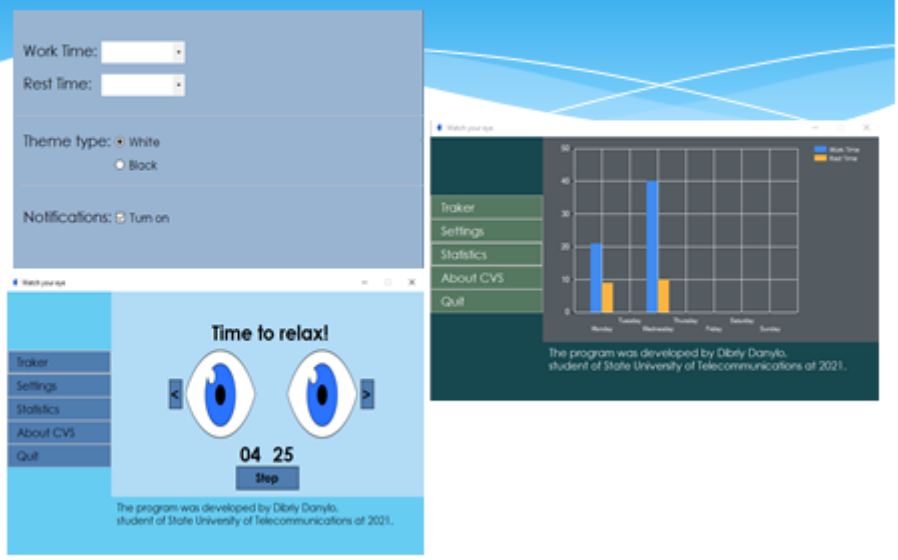
Використані технології

- Мова програмування – C#.
- Середа розробки – Microsoft Visual Studio.
- Технологія керування даними – автономний рівень ADO.NET.
- Інтерфейс програмування додатків – Microsoft Windows Forms.
- Сервіси для проведення тестування – Jira та TestRail.

Принцип взаємодії користувача та системи



Реалізоване програмне забезпечення



АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Основні положення і результати бакалаврської роботи доповідались і обговорювались на науково-практичних конференціях:

- Дібрій Д.А. Розробка ПЗ для автоматизації процесу зберігання здоров'я очей користувача під час роботи за ПК // Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інфокомунікаційних технологія» Збірник тез. 12.02.2021, ДУТ, м. Київ — К.: ДУТ, 2021. — С. 64.
- Дібрій Д.А. Розробка десктопного ПЗ для зберігання здоров'я очей користувача на мові програмування C# // Всеукраїнська науково-технічна конференція «Сучасний стан та перспективи розвитку IOT» Збірник тез. 9.04.2021, ДУТ, м. Київ — К.: ДУТ, 2021. — С. 37.
- Дібрій Д.А. Актуальність створення ПЗ для зберігання здоров'я очей користувача // Всеукраїнська науково-технічна конференція «Сучасний стан та перспективи розвитку IOT» Збірник тез. 9.04.2021, ДУТ, м. Київ — К.: ДУТ, 2021. — С. 39.

Висновки

Було проаналізовано недоліки аналогічного БЗ та створено десктопне програмне забезпечнення для збереження здоров'я користувачів під час роботи за ПК, яке надає більш широкий функціонал, що роблить його привабливішим у очах користувача.

Створене прикладне програмне забезпечнення має велику соціально-практичну значущість, оскільки допомагає людям які працюють за екранними пристроями та звичайним користувачам зберегти свій зір та покращити загальне самопочуття.

ДЯКУЮ ЗА УВАГУ!