Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА

на тему:

Безпечний органайзер з аудіосповіщенням для навчання

Виконав студент 4 курсу, групи 1КІ-18б
спеціальності 123 — Комп'ютерна інженерія

Вторук О.В

Керівник к.т.н., доц. каф. ОТ

Савицька Л. А.

"15" 06 2022 р.

Рецензист к.т.н., доц. каф. МБІС

Шиян А. А.

"15" 06 2022 р.

Допущено до захисту

Зав. кафедри ОТ

д.т.н., проф. Азаров О. Д.

"16" 06 2022 р.

ВНТУ 2022

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітній рівень — Бакалавр
Спеціальність — 123 Комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри обчислювальної техніки

О.Д. Азаров

"09" 02. 2022 р.

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ

студенту **Вторуку Олександру Володимировичу**

1 Тема роботи «Безпечний органайзер з аудіосповіщенням для навчання» керівник роботи Савицька Людмила Анатоліївна к.т.н., доцент, затверджено наказом вищого навчального закладу від **24.03.22** року **№ 66**

2 Строк подання студентом роботи **15.06.2022 р.**

3 Вихідні дані до роботи: Стандарти, монографії, підручники та наукові статті по темі. Існуюче програмне забезпечення, яке стосується теми бакалаврської дипломної роботи.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз підходів до розробки віконних додатків, розробка структури та алгоритмів роботи органайзеру, програмна реалізація та тестування програми, висновки.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): діаграма класів.

6 Консультанти розділів бакалаврської дипломної роботи.

Таблиця 1— Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
I	Савицька Л. А к.т.н., доцент каф. ОТ		
II	Савицька Л. А к.т.н., доцент каф. ОТ		
III	Савицька Л. А к.т.н., доцент каф. ОТ		

7 Дата видачі завдання **08.02.2022 р.**

8 Календарний план виконання БДР приведений в таблиці 2.

Таблиця 2 — Календарний план

№	Назва та зміст етапу	Термін виконання		Примітка
		Початок	Закінчкення	
1	Визначення напрямку бакалаврської роботи, формулювання теми	09.02.22	16.02.22	**Виконав**
2	Аналіз предметної області обраної теми	17.02.22	24.02.22	**Виконав**
3	Розробка алгоритму роботи	29.02.22	21.03.22	**Виконав**
4	Написання бакалаврської роботи на основі розробленої теми	22.03.22	20.04.22	**Виконав**
5	Передзахист бакалаврської дипломної роботи	21.04.22	15.05.22	**Виконав**
6	Виправлення, уточнення, корегування бакалаврської дипломної роботи	16.05.22	05.06.22	**Виконав**
7	Захист бакалаврської дипломної роботи	16.06.22	16.06.22	**Виконав**

Студент (підпис) Вторук О. В.

Керівник (підпис) к.т.н., доц. Савицька Л. А.

АНОТАЦІЯ

Вторук О. В. Безпечний органайзер з аудіосповіщенням для навчання. Бакалаврська дипломна робота зі спеціальності 123 – комп’ютерна інженерія, освітня програма – комп’ютерна інженерія. Вінниця: ВНТУ, 2022. с.

Бакалаврська робота складається з 41 сторінок А4 на яких є 16 рисунків 1 таблиць, список використаних джерел містить 9 найменувань.

Метою роботи є розробка безпечного органайзера з аудіосповіщенням для навчання в середовищі Visual Studio 2019, який дозволить не забувати про пари та екзамени.

У загальній частині роботи розглянено підходи до розробки віконних додатків та актуальні середовища розробки. У розрахунково конструкторській частині виконано проектування безпечного органайзера з аудіосповіщенням для навчання. У технологічній частині, розроблено та протестовано розроблену програму.

Ключові слова: органайзер, подія, планування, шифрування, AES256, навчання, аудіосповіщення.

ABSTRACT

The bachelor's thesis consists of 41 A4 pages with 16 figures and 1 table, the list of used sources contains 9 items.

The aim of the work is to develop a secure organizer with audio notification for learning in Visual Studio 2019, which will not forget about couples and exams.

In the general part of the work approaches to the development of window applications and current development environments are considered. In the calculation and design part, the design of a safe organizer with audio notification for training was performed. In the technological part, the developed program is developed and tested.

Keywords: organizer, event, planning, encryption, AES256, training, audio notification.

ЗМІСТ

ВСТУП

У нас все більше справ і ми нерідко забуваємо щось зробити. Для цього людство придумало безліч методів. В давнину в римі були раби, котрі нагадували своїм хазяїнам про важливі справи. Пізніше люди почали робити замітки у блокнотах та щоденниках. Ще пізніше до нас прийшли стікери, якими обліплювали все, від робочого столу до холодильника. А зараз, у часи комп'ютерної ери, є цифрові нагадувачі, органайзери, які допоможуть не забути про свій графік та важливі справи.

Дуже важливо, в даному питанні залишатися приватним. Зазвичай люди не діляться своїми планами з незнайомцями, а є такі справи в які не варто посвячувати і рідних. Тому варто додати шифрування та вхід по паролю до органайзеру і ми отримаємо захищений планувальник подій.

Тому **актуальність** даної роботи в тому, що розроблена програма буде безпечна. Дана безпека забезпечується шифруванням одним з актуальних на даний час шифрів.

Метою роботи є розробка безпечного органайзера з аудіосповіщенням для навчання в середовищі Visual Studio_2019, який дозволить не забувати про пари та екзамени.

Задачі дослідження бакалаврської роботи:

— проаналізувати методи та технології розробки додатків ООП;

— проаналізувати існуючі аналоги;

— проаналізувати та обрати засоби реалізації системи;

— виконати моделювання;

— виконати розробку віконного додатку за допомогою мови програмування C#;

— протестувати розроблений додаток.

Об'єкт дослідження — процес створення безпечного органайзера з аудіосповіщенням для навчання.

Предмет дослідження — безпечний органайзер з аудіосповіщенням для навчання.

Методи дослідження бакалаврської роботи: у роботі використано принципи об’єктно-орієнтованого програмування мовою C# для реалізації підходу.

Апробація результатів бакалаврської роботи: зроблено доповідь на L науково–технічній конференції підрозділів Вінницького національного технічного університету.

Практичне значення отриманих результатів полягає в можливості використовування розробленої системи в повсякденні звичайних студентів та людей.

1. АНАЛІЗ ПІДХОДІВ ДО РОЗРОБКИ ВІКОННИХ ДОДАТКІВ

1.1. Огляд існуючих мов програмування

Як відомо комп'ютер сприймає команди в двійковому коді, але програми на ньому писали в основному до середини минулого століття. Набагато спрощувало створення програм використання мов програмування.

Мову Асемблер люди почали використовувати з 1950 року. Ця мова першою дозволила відображати двійковий код в більш зручній для людини формі: у вигляді букв або укорочених слів, які приблизно позначали сутність команди. Вміючи розбиратися в кодах асемблера, можна знаходити помилки в програмах створених навіть за допомогою інших мов [1].

Також на асемблері пишуть коли потрібно домогтися високої швидкодії, так на даній мові написано ColibriOS, операційну систему яка поміщається на дискету і працює навіть на зовсім старому “залізі” [2].

Ada — була розроблена за підтримки Міноборони США в 1978 році в результаті конкурсу на якому виграла група програмістів компанії Honeywell. Цікаво, що всі мови, які дійшли до останніх турів цього конкурсу, були засновані на Паскалі. У зв'язку з цим Аду можна попередньо охарактеризувати як розвинений Паскаль [3].

Названа по імені першої жінки програмістки Ади Лавлейс. Кінцева специфікація розроблена до 1983 року. Поза військових проектів мова широкого поширення не отримала [4].

Мова програмування Сі була розроблена в 1972 році Деннісом Рітчі, в компанії Bell Laboratories. Назва походить від номера проекту лабораторії ("A", "B", C "...). Спочатку ця мова замислювався як проміжна між мовами високого і низького рівнів, але продуктивність і компактність коду + переваги структурного мови, незважаючи на складність навчання зробили Сі найпопулярнішою мовою.

У 1980 році компанія випускає нову мову побудований на основі C — C ++. За словами розробника, Бьярн Страуструпа вона повинна «довершити написання хороших програм і зробити цей процес найбільш приємним для кожного окремо

взятого програміста ». Назва С ++ було придумано замість початкового «С з класами» [5].

Персі версії ОС Windows були написані на С. Крім «Вікон» на Сі написані такі відомі програми як Outlook, Opera, 1С.

Java Створена компанією Sun Microsystems. система розробки Java безпечна і високопродуктивна. Java — об'єктно-орієнтована мова, зручна і надійна в використанні завдяки таким своїм перевагам, таким як багатозадачність, підтримка протоколів Internet і крослатформеність. Java – це мова, що інтерпретується, і кожна Java-програма компілюється для гіпотетичної машини, званої JVM.

Результатом такої компіляції є байт-код Java, який в свою чергу може виконуватися на будь-якій операційній системі за умови наявності там JVM, яка інтерпретує байт-код в реальний машинний код конкретної машини.

Мова Java є об'єктно-орієнтованою і поставляється з досить об'ємною бібліотекою класів. Бібліотеки класів Java значно спрощують розробку додатків, надаючи в розпорядження програміста потужні засоби вирішення поширених завдань [6].

На Java написаний двигун відомого емулятора «Іл-2: Штурмовик».

C# — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (належить Microsoft).

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато від своїх попередників — мов C++, Object Pascal, Модула і Smalltalk — C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад, мова C#, на відміну від C++, не передбачає множинне успадкування класів [7].

Символ # у назві мови можна інтерпретувати і як дві пари плюсів ++, що натякають на новий крок у розвитку мови порівняно з C++ (подібно до кроку від C до C++), і як музичний символ дієз, разом з буквою C, що становить в англійській мові назву ноти до-дієз. Останнє й дало назву мові. Попри те, що символ # (октоторп) насправді є символом для позначення номера на більшості клавіатур і відрізняється від символу дієз ♯ (Unicode U+266F), Microsoft, як автор мови, неодноразово зверталася до своїх клієнтів з проханням прийняти таку стилізацію [8].

Microsoft вирішили, користуючись своїм панівним становищем на ринку, створити свій власний аналог Java — мову, для якої корпорація буде повноцінним власником. Ця новостворена мова отримала назву C#. Вона успадкувала від Java концепції віртуальної машини (середовище .NET), байт-коду (MSIL) і більшої безпеки вихідного коду програм, також було враховано досвід використання програм на Java.

Нововведенням C# стала можливість легшої взаємодії, порівняно з мовами-попередниками, з кодом програм, написаних на інших мовах, що є важливим при створенні великих проєктів. Якщо програми на різних мовах виконуються на платформі .NET, .NET бере на себе клопіт щодо сумісності програм (тобто типів даних, за кінцевим рахунком) [9-11].

Станом на сьогодні C# визначено флагманською мовою корпорації Microsoft, бо вона найповніше використовує нові можливості .NET. Решта мов програмування, хоч і підтримуються, але визнані такими, що мають спадкові прогалини щодо використання .NET.

1.2 Вибір IDE для розробки

Існує кілька середовищ розробки для мови програмування C#.

1.2.1 Огляд Visual Studio Code

Visual Studio Code — засіб для створення, редагування та зневадження сучасних веб додатків і програм для хмарних систем. Visual Studio Code є безкоштовною і доступною у версіях для платформ Windows, Linux та OS X.

Компанія Microsoft представила Visual Studio Code у квітні 2015 на конференції Build 2015. Це середовище розробки стало першим кросплатформовим продуктом у лінійці Visual Studio.

За основу для Visual Studio Code використовуються напрацювання вільного проєкту Atom, що розвивається компанією GitHub. Зокрема, Visual Studio Code є надбудовою над Atom Shell, що використовує браузерний рушій Chromium і Node.js. Примітно, що про використання напрацювань вільного проєкту Atom і на сайті Visual Studio Code, і в пресрелізі, і в офіційному блозі не згадується [12].

1.2.2 Огляд SharpDevelop

SharpDevelop — середовище розробки для C++, C#, Visual Basic .NET, Boo, IronPython, IronRuby, F#.Використовується як альтернатива Visual Studio .NET. Існує також форк на Mono/GTK+ — MonoDevelop. Остання версія вийшла в квітні 2016 року. На жаль на даний час розробка зупинена.

SharpDevelop 2.0 надає інтегрований налагоджувач, який використовує власні бібліотеки і взаємодіє з середовищем .NET через COM Interop.

Хоча SharpDevelop 2.0 (як і VS) використовує файли проекту у форматі MSBuild, воно як і раніше може використовувати компілятори від .NET Framework 1.0 і 1.1, а також від Mono.

Остання версіся підтримує версії .NET Framework від 2.0 до 4.5.1 і .NET Compact Framework 2.0 і 3.5

1.2.3 Огляд Rider

Rider — кроссплатформент інтегрованt середовище розробки програмного забезпечення для платформи .NET, що розробляється компанією JetBrains. Підтримуються мови програмування C#, VB.NET та F#.

Проект анонсовано у січні 2015 року. В його основі лежить інший продукт JetBrains — ReSharper — одна з найкращих надбудов для Visual Studio. Середовище підтримує платформи .NET Framework, .NET та Mono. Rider працює на операційних системах Windows, MacOS, Linux.

Реліз відбувся 3 серпня 2017.

1.2.4 Огляд MS Visual Studio

Microsoft Visual Studio — лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення та інші інструменти. Дані продукти дозволяють розробляти як консольні програми, так і ігри та програми з графічним інтерфейсом, у тому числі з підтримкою технології Windows Forms, UWP а також веб-сайти, веб-програми, веб-служби як в рідному, так і в керованому кодах всіх платформ, підтримуваних Windows, Windows Mobile, Windows CE, .NET Framework, .NET Core, .NET, MAUI, Xbox, Windows Phone .NET Compact Framework та Silverlight. Після придбання компанії Xamarin корпорацією Microsoft з'явилася можливість розробки IOS та Android програм.

Visual Studio включає редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторингу коду. Вбудований налагоджувач може працювати як налагоджувач рівня вихідного коду, так і відладчик машинного рівня. Інші вбудовані інструменти включають редактор форм для спрощення створення графічного інтерфейсу програми, веб-редактор, дизайнер класів і дизайнер схеми бази даних. Visual Studio дозволяє створювати та підключати сторонні доповнення (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду (як, наприклад, Subversion та Visual SourceSafe), додавання

нових наборів інструментів (наприклад, для редагування та візуального проектування) коду предметно-орієнтованими мовами програмування) або інструментами для інших аспектів процесу розробки програмного забезпечення (наприклад, клієнт Team Explorer для роботи з Team Foundation Server) [13].

1.3 Поняття ООП

ООП — це об’єктно орієнтоване програмування. Якщо простими словами, то це підхід до програмування, який намагається представити об’єкти реального світу в об’єктах певних класів. Клас – це сукупність полів та методів. Якщо розглядати залежність класу та об’єкту, то вона така ж як у типу та змінної.

ООП тримається на трьох китах – наслідування, інкапсуляція та поліморфізм.

Наслідування — механізм, що дозволяє класам нащадкам успадковувати поля батьківських класів. Цей механізм суттєво зменшує обсяг коду, оскільки не потрібно дублювати одне і те ж у всіх класах.

Наприклад у нас є клас Рослина, з полями назва та вік. Якщо ми створимо клас Дерево та успадкуємо його від класу рослина, то ми можемо добавити поле висоти, а поля назва та вік успадкуються з батьківського класу.

Варто відмітити, що C# не дозволяє множинного наслідування, оскільки це може призводити до колізій. Якщо є потреба множинного наслідування то можна використати наслідування від інтерфейсу. Кожен клас може бути нащадком безлічі інтерфейсів. Проте, кожен інтерфейс повинен бути повністю реалізований в класі.

Інкапсуляція – механізм, що дозволяє обмежувати доступ до певних членів. Якщо розглядати на прикладі мови програмування C#, то є 5 модифікаторів доступу [8,9]

- public – доступ можливий звідки завгодно;
- private – доступ лише в межах даного класу;
- protected – доступ з поточного чи наслідуваного класу;

— internal – доступ з поточної збірки;

— protected internal – доступ з поточної збірки та наслідуваних класів;

— private protected – лише наслідувані класи з поточної збірки.

Поліморфізм — механізм, суть якого в тому, що один і той же фрагмент коду, може працювати з різними типами даних. Сюди можна віднести абстрактні класи та віртуальні функції а також шаблонні функції, однак основна суть поліморфізму в перетворенні дочірніх класів до батьківських шляхом утинання частини даних. Зазвичай таке перетворення потребує явного перетворення типів.

Концепція віртуальної функції вирішує наступну проблему [14]:

У ООП, якщо клас-нащадок наслідується від базового класу, об'єкт екземпляр класу-нащадка може використовуватись або як екземпляр батьківського класу (бути приведеним до батьківського класу), або як екземпляр класу-нащадка. Якщо у класі-нащадку є функції, що перекривають (мають таку ж сигнатуру) функції із батьківського класу, то поведінка при виклику таких методів (при використанні даного об'єкта як екземпляра батьківського класу) є невизначеною[15-18].

Відмінність між віртуальністю і невіртуальністю функцій вирішує цю невизначеність. Якщо функція описана як віртуальна у базовому класі, тоді буде викликана функція із класу нащадка (якщо така існує). Якщо вона не віртуальна, тоді — із батьківського класу[19-20].

Якщо підсумувати, то поліморфізм – це механізм що дозволяє оперувати класами нащадками так, наче вони батьківські, починаючи від обробки даних і закінчуючи приведенням до типу [21].

1.4 Аналіз існуючих органайзерів

Наразі є безліч органайзерів, ті що одразу приходять на думку:

— Google Calendar

— Weeek

— Todoist

— Remember The Milk

Розглянемо їх детальніше.

Google Calendar — це безкоштовний органайзер. Дозволяє створювати події, задачі та нагадування. Мабуть найпоширеніший органайзер у світі оскільки є веб версія і доступ можливий з усіх девайсів.

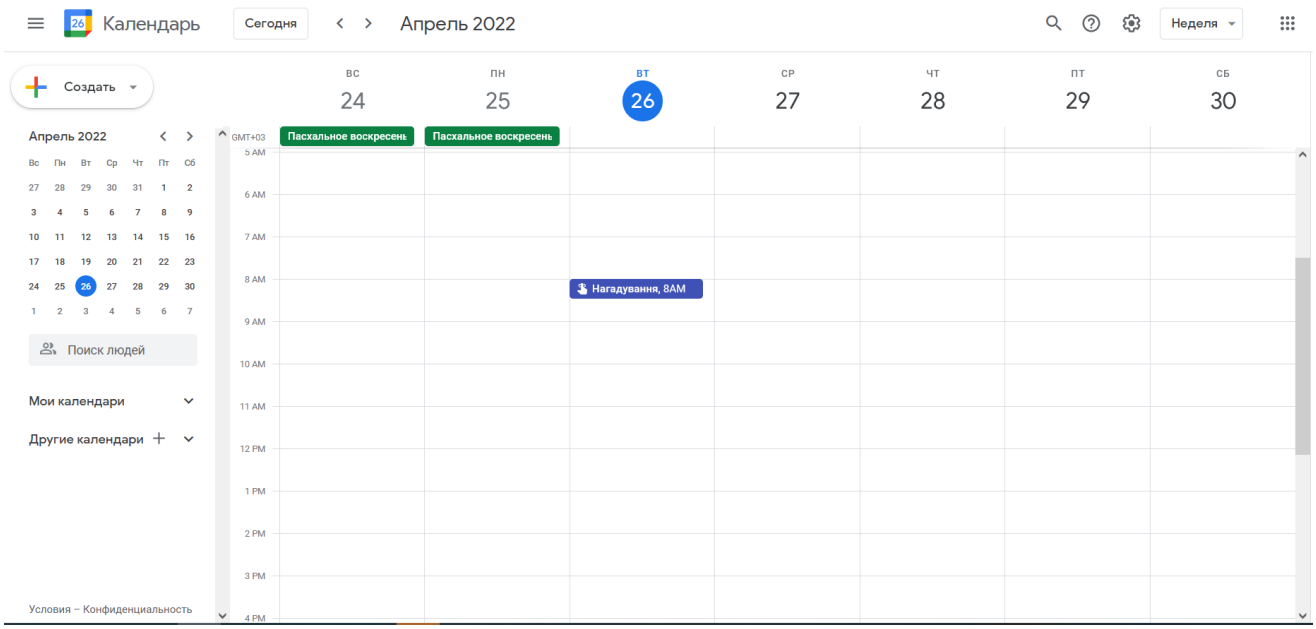

Рисунок 1.1 — Google Calendar

Weeek — це продумана система управління завданнями та проектами, заснована на канбан-методології. Гнучка структура (воркспейси → проекти → дошки). Багатоступінчасті завдання. Вбудований тайм-трекер та Pomodoro-таймер. Мобільні програми на iOS та Android. Є безкоштовна версія з невеликими обмеженнями щодо кількості проектів та учасників команди.

Todoist — це простий і гарний онлайн планувальник, який підтримує практичні всі мобільні та настільні платформи. В т.ч. працює в поштових програмах та сервісах. Дозволяє спільно працювати над завданнями, отримувати нагадування та оповіщення.

Remember The Milk — це плагін до MS Outlook, який дозволяє синхронізувати завдання з Outlook з вашим обліковим записом в онлайн сервісі

для управління завданнями Remember the Milk. Як відомо, Remember the Milk підтримує офлайн доступ за допомогою Google Gears, так що цей плагін

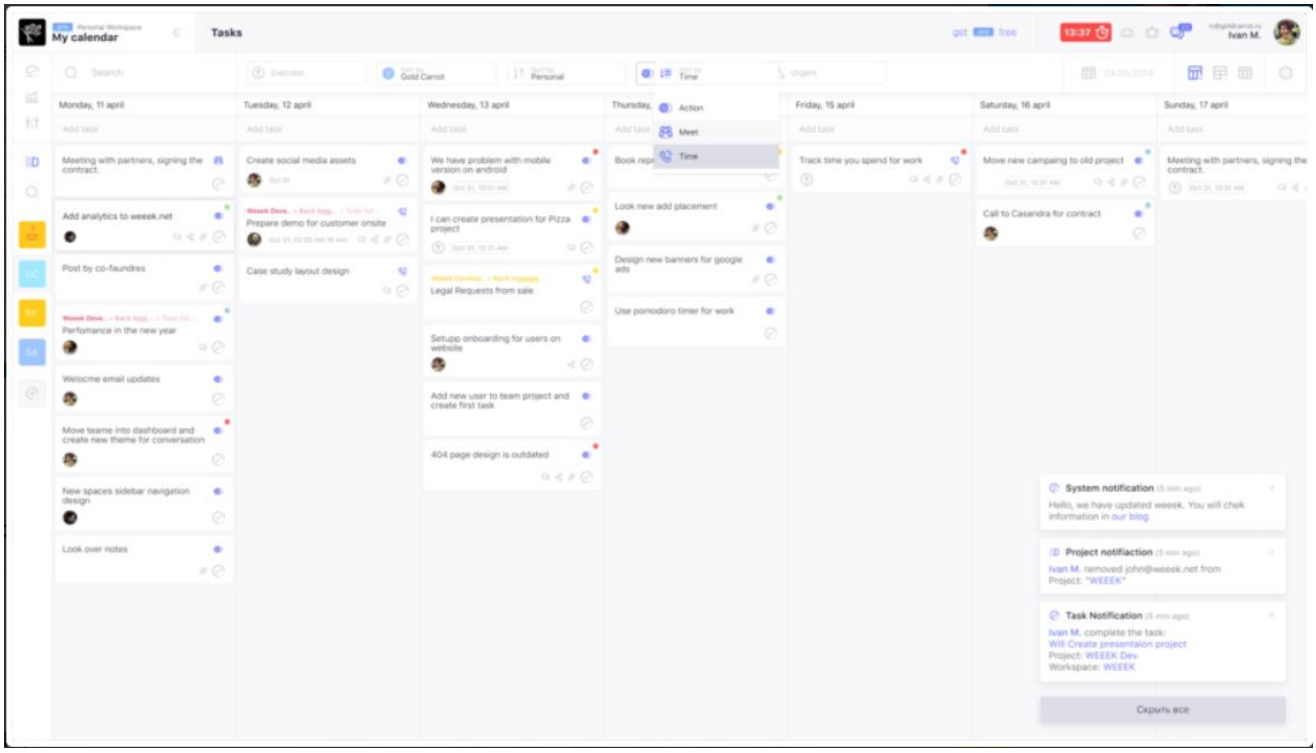

Рисунок 1.2 — Weeek.

насамперед буде цікавий користувачам Outlook, які хочуть отримати доступ до своїх завдань в онлайні або обмінюватися завданнями зі своїми колегами без необхідності встановлення сервера Exchange. Новий плагін працює за допомогою Remember the Milk API і є маленьким тулбаром на панелі інструментів Outlook. Синхронізувати завдання можна лише вручну – натисканням кнопки. Слід сказати, що синхронізація займає досить багато часу. При синхронізації категорії Outlook перетворюються на категорії списків RTM. Плагін розповсюджується безкоштовно.

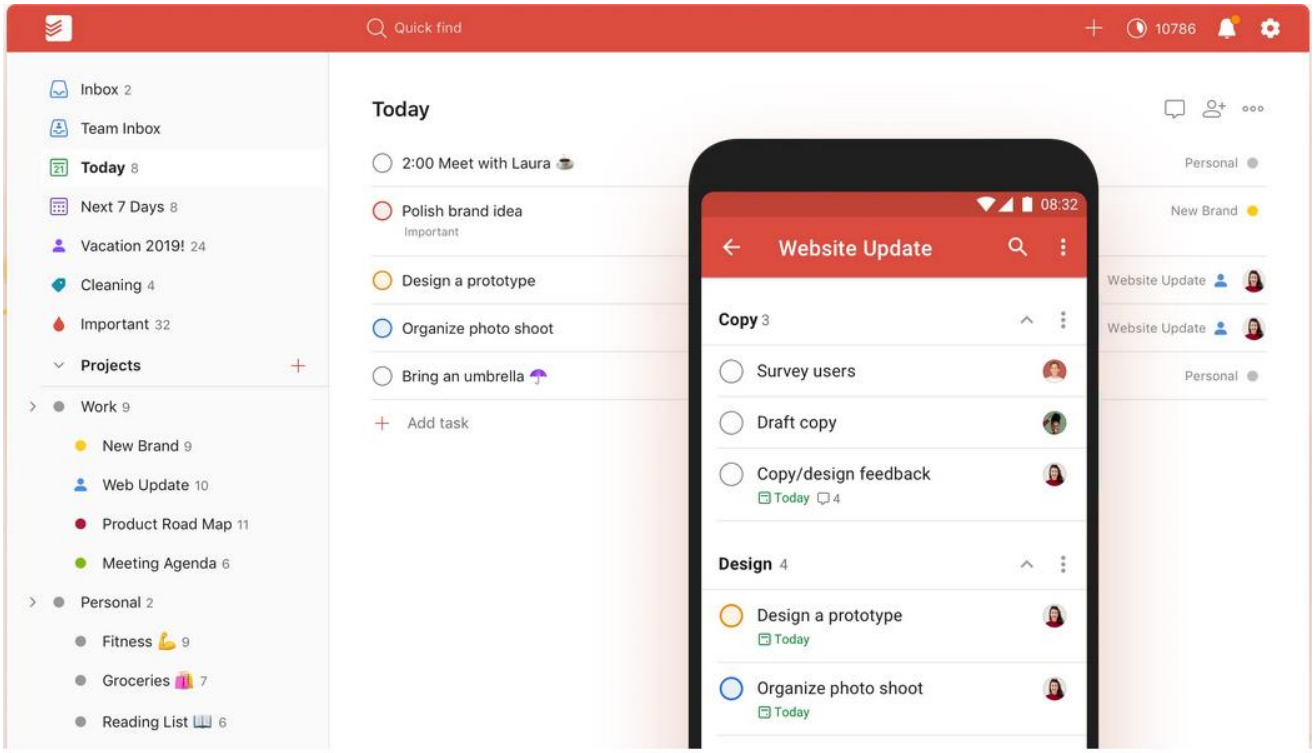

Рисунок 1.3 — Todoist.

2. РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ РОБОТИ ОРГАНАЙЗЕРУ

2.1 Розробка інтерфейсу користувача

Інтерфейс користувача (UI) було розроблено з використанням технології WPF. Дана технологія являється передовою в розробці користувацьких інтерфейсів. Її основна відмінність від свого попередника Winforms в плані проектування інтерфейсів в тому, що всі елементи можна дуже налаштовувати дуже гнучко, створювати користувацькі контроли та налаштовувати елементи контролів на кшталт границь кнопок чи позиціонування кутиків.

При запуску програми відкривається форма входу.

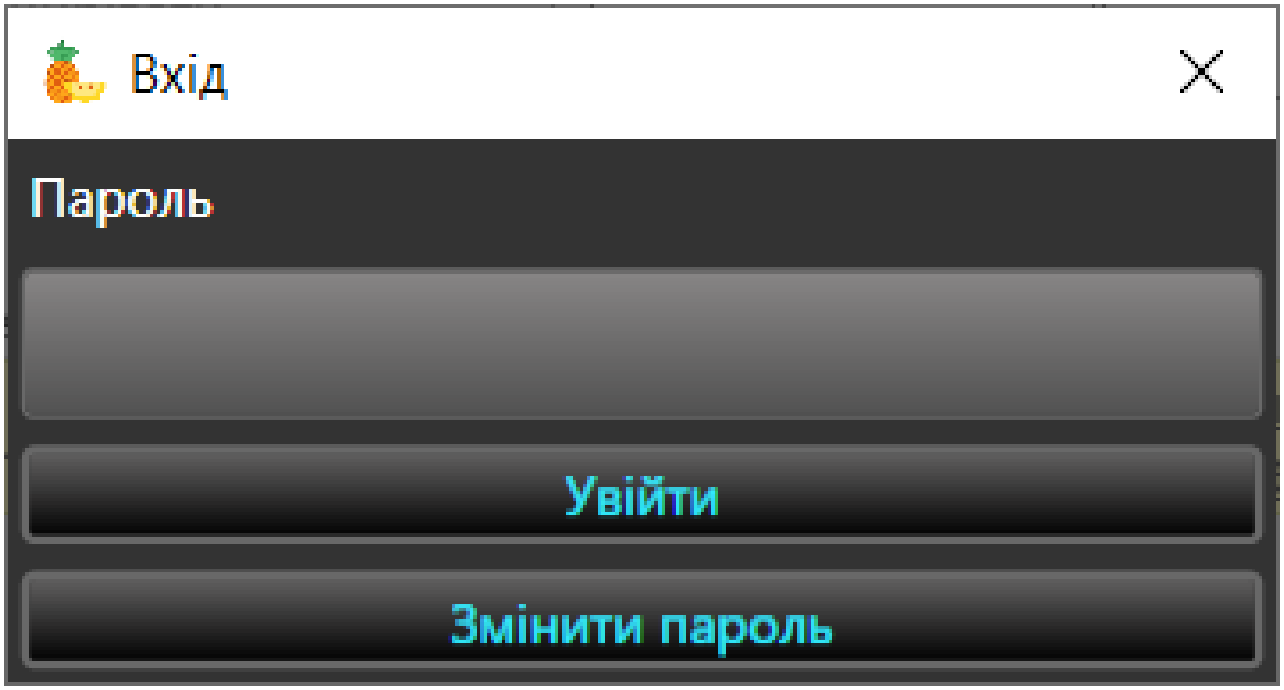

Рисунок 2.1 — Форма входу

Лістинг 2.1. Розмітка форми входу

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="auto"></RowDefinition>
        <RowDefinition Height="auto"></RowDefinition>
        <RowDefiniion Height="*"></RowDefinition>
```

Продовження лістингу 2.1.

```
            <RowDefinition Height="*"></RowDefinition>
        </Grid.RowDefinitions>
        <Grid Grid.Row="0">
            <Label Foreground="White">Пароль</Label>
        </Grid>
        <Grid Grid.Row="1" Margin="3">
            <PasswordBox Name="Pass" PasswordChar="*"></PasswordBox>
        </Grid>
        <Grid Grid.Row="2" Margin="3">
            <Button Click="Button_Click">Увійти</Button>
        </Grid>
        <Grid Grid.Row="3" Margin="3">
            <Button Click="Button_Click_1">Змінити пароль</Button>
        </Grid>
    </Grid></Window>
```

По замовчуванню пароль 123, проте за потреби його можна змінити. Для зміни паролю потрібно клікнути на кнопку “Змінити пароль”. З’явиться форма зміни паролю (див рис. 2.2).

Якщо пароль ввести неправильно тричі, то дана форма закриється. Це реалізовано аби захиститися від брутфорсу, тобто повного перебору паролів грубою силою.

Все інше зробить криптобезпека алгоритму AES з 256 бітним ключем, який є стандартом шифрування США вже більше 20 років і досі вважається достатньо надійним аби ним захищати важливі документи в тому числі й ті які містять державні таємниці. Розмітка сторінки показана на лістингу 2.1.

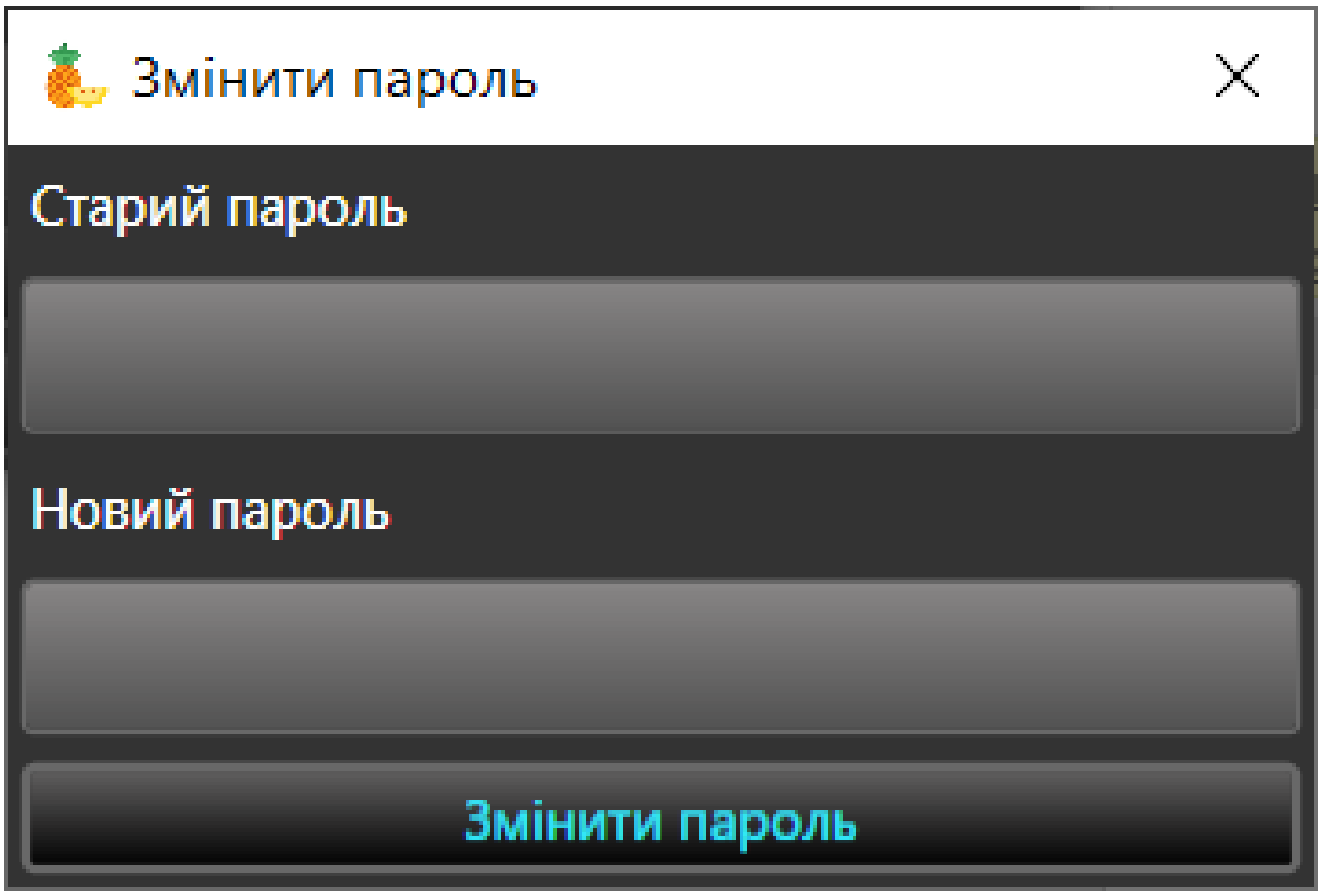

Рисунок 2.2 — Форма зміни паролю

Дуже важливо не забути пароль, оскільки при втраті паролю всі дані залишаться зашифрованими і на їх дешифрування знадобиться значний час.

Якщо ви забули свій пароль ви зможете його змінити. Кількість спроб на зміну паролю необмежена.

Якщо при вході пароль було введено вірно відкриється головна форма програми. На ній розміщено календар та дві кнопки з музикою та налаштуваннями. Розмітка даної сторінки показана на рис 2.2.

Лістинг 2.2. Розмітка форми зміни паролю

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="auto"></RowDefinition>
        <RowDefinition Height="auto"></RowDefinition>
        <RowDefinition Height="auto"></RowDefinition>
```

Продовження лістингу 2.2.

```
            <RowDefinition Height="auto"></RowDefinition>
            <RowDefinition Height="*"></RowDefinition>
        </Grid.RowDefinitions>
        <Grid Grid.Row="0">
            <Label Foreground="White">Старий пароль</Label>
        </Grid>
        <Grid Grid.Row="1" Margin="3">
            <PasswordBox Name="OldPass" PasswordChar="*"></PasswordBox>
        </Grid>
        <Grid Grid.Row="2">
            <Label Foreground="White">Новий пароль</Label>
        </Grid>
        <Grid Grid.Row="3" Margin="3">
            <PasswordBox Name="NewPass" PasswordChar="*"></PasswordBox>
        </Grid>
        <Grid Grid.Row="4" Margin="3">
            <Button Click="Button_Click">Змінити пароль</Button>
        </Grid>
    </Grid>
```

При кліку на кнопку налаштувань відкриється форма налаштувань. На даній формі є основні налаштування такі як час за який буде нагадано про подію. Тобто якщо пара починається о 10 00, а вказано 15 хв, то сповіщення з’явиться о 9 45.

Час відтворення аудіо. Це час у секундах, який лунатиме музика. Немає сенсу включати музику навічно, оскільки якщо користувач не біля ПК, то це лише буде садити акумулятор та дратувати оточуючих. А також час показу сповіщень. Це час протягом якого будуть показані сповіщення віндовс. Саме цей параметр є сенс ставити на подовше, оскільки якщо користувач біля ПК, він просто згорне

сповіщення, а якщо відійшов, то саме сповіщення йому нагадає що у нього щось заплановано.

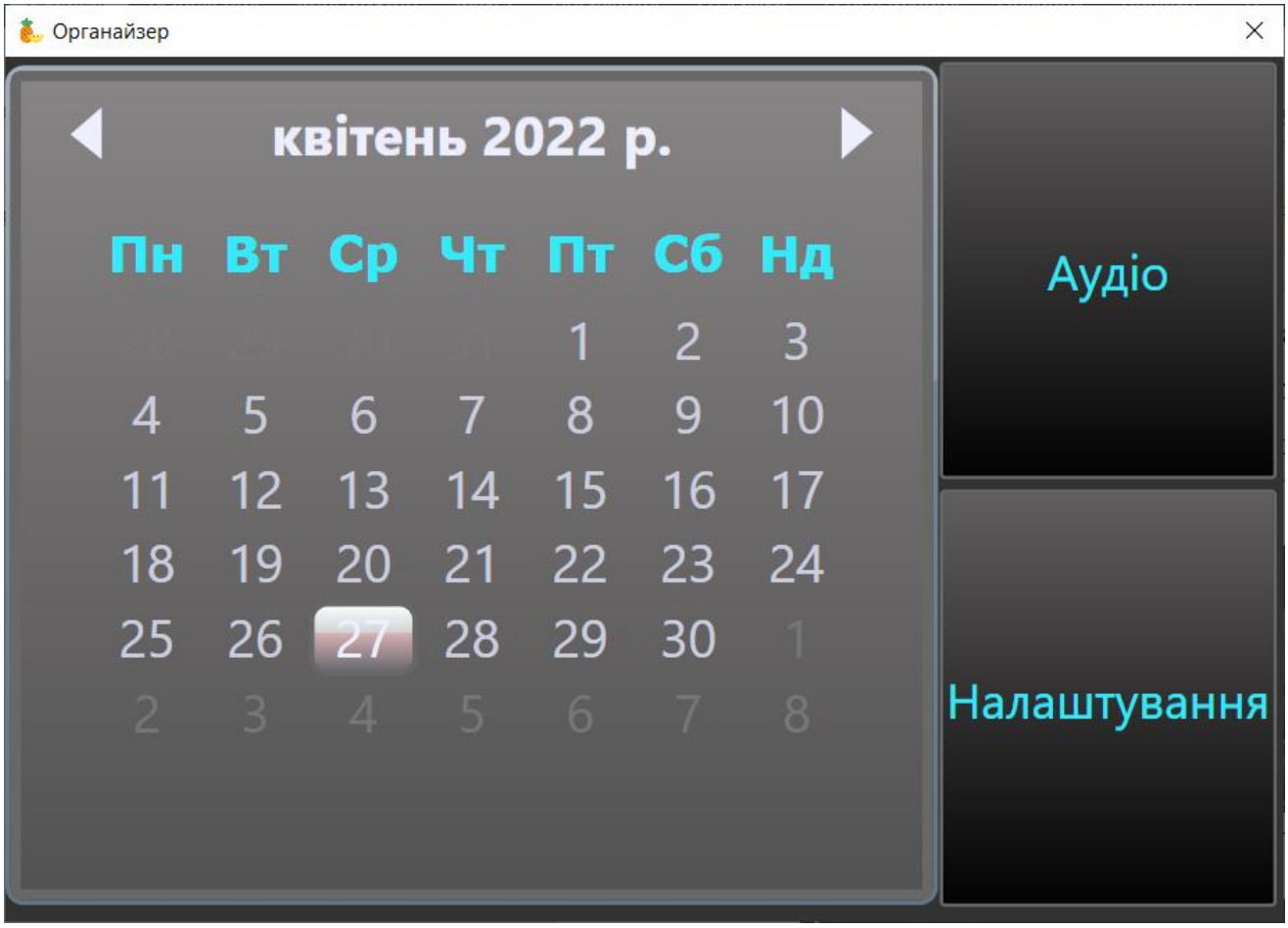

Рисунок 2.3 —Головна форма

Також на даній форма можна розмістити інші налаштування, які будуть добавлені в майбутньому.

Лістинг 2.3. Головна форма

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="535"></ColumnDefinition>
        <ColumnDefinition Width="200"></ColumnDefinition>
    </Grid.ColumnDefinitions>
    <Grid Grid.Column="0" Margin="0,-10,0,0">
```

Продовження лістингу 2.3.

```
            <StackPanel HorizontalAlignment="Left" VerticalAlignment="Top">
              <Calendar Name="MainCalendar"
MouseDoubleClick="MainCalendar_MouseDoubleClick">
                <Calendar.RenderTransform>
                  <ScaleTransform ScaleX="3" ScaleY="3"/>
                </Calendar.RenderTransform>
              </Calendar>
            </StackPanel>
          </Grid>
          <Grid Grid.Column="1">
            <Grid.RowDefinitions>
              <RowDefinition Height="auto"></RowDefinition>
              <RowDefinition Height="auto"></RowDefinition>
              <RowDefinition Height="auto"></RowDefinition>
              <RowDefinition Height="auto"></RowDefinition>
            </Grid.RowDefinitions>
            <Grid Grid.Row="0">
              <Button Height="240" FontSize="28" Margin="3"
Click="Button_Click">Аудіо</Button>
            </Grid>
            <Grid Grid.Row="1">
              <Button Height="240" FontSize="28" Margin="3"
Click="Button_Click_1">Налаштування</Button>
            </Grid>
          </Grid>
        </Grid>
```

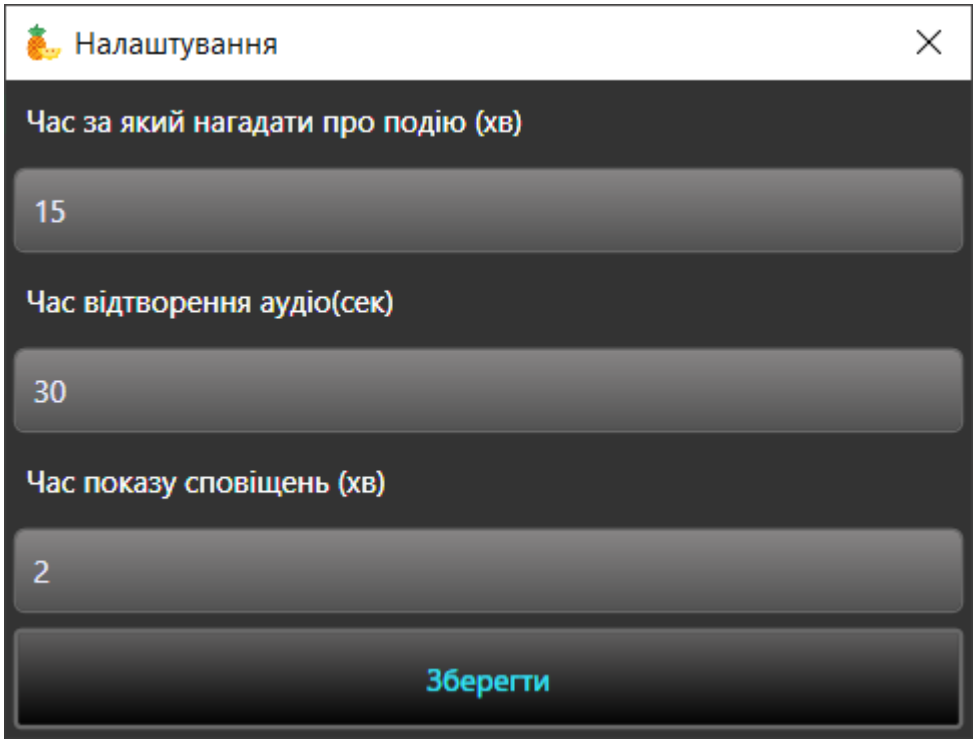

Рисунок 2.4 — Форма налаштувань

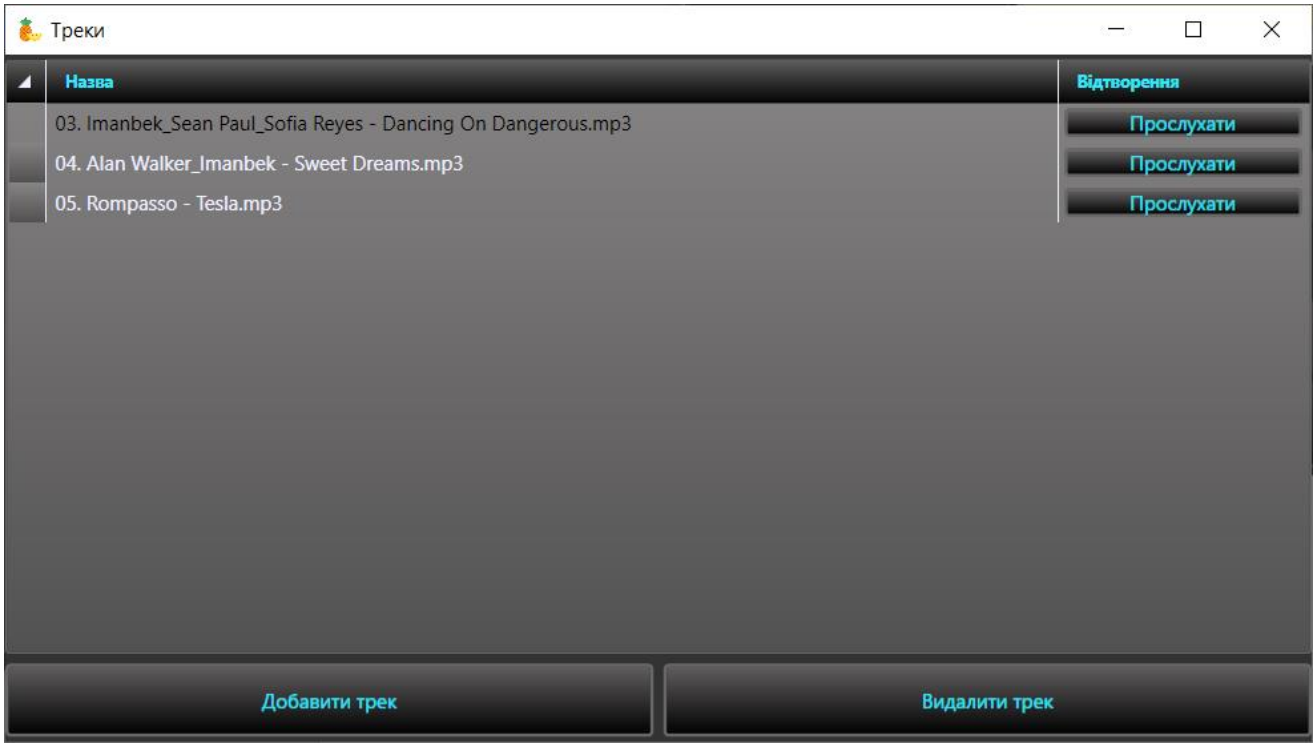

Рисунок 2.5 — Форма аудіо

Даний органайзер може відтворювати музику в mp3 форматі. Для додавання треку потрібно натиснути однойменну кнопку та обрати файл у потрібному

форматі. Також його можна одразу ж і прослухати використовуючи кнопки на формі.

Для перегляду розкладу на день, потрібно на головній формі двічі клікнути по потрібній даті.

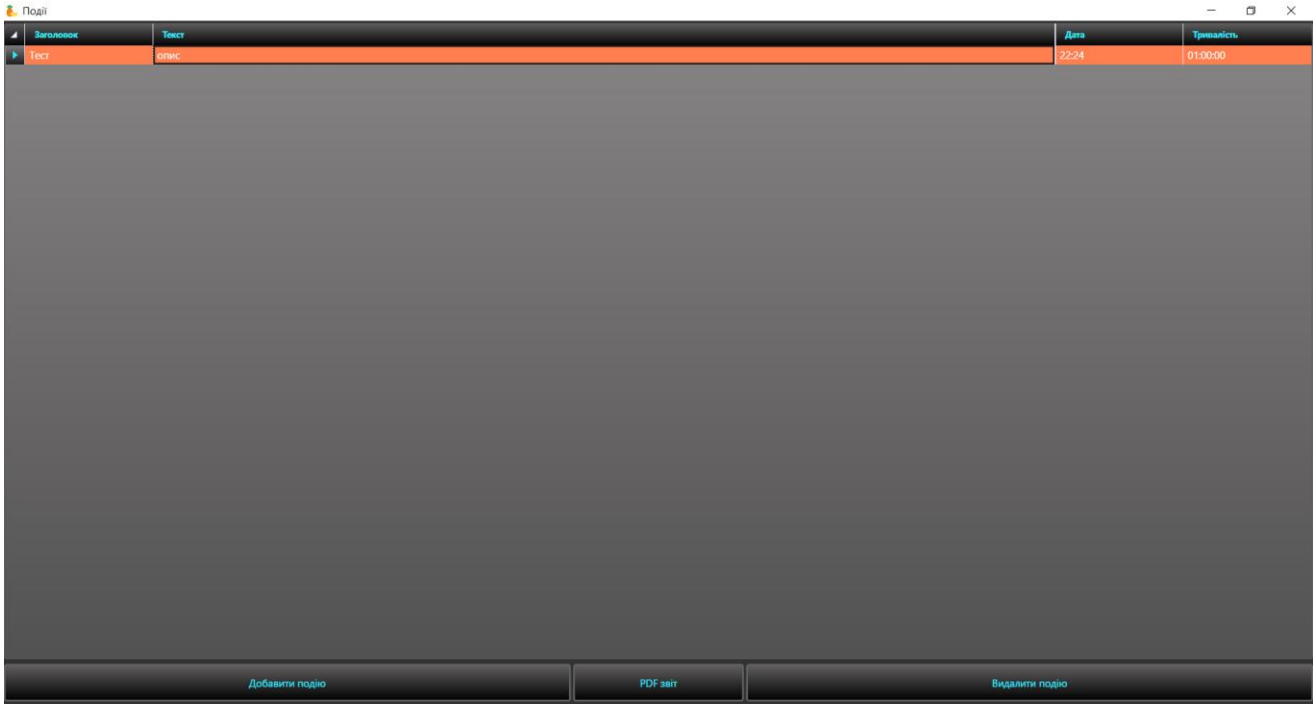

Рисунок 2.6 — Форма розкладу

За потреби додати подію потрібно клікнути на кнопку “Добавити подію”. Після цього з’явиться форма події (див рис. 2.7).

Якщо ж потрібно редагувати подію, то потрібно двічі клікнути мишею на рядок події і відкриється та ж форма (див рис. 2.7), проте з заповненими відповідними даними. В тому числі датою та кольором, адже програма не бачить різниці між текстовим та іншими форматами даних.

Також на даній формі присутня кнопка створення PDF звіту який міститиме всі події дня.

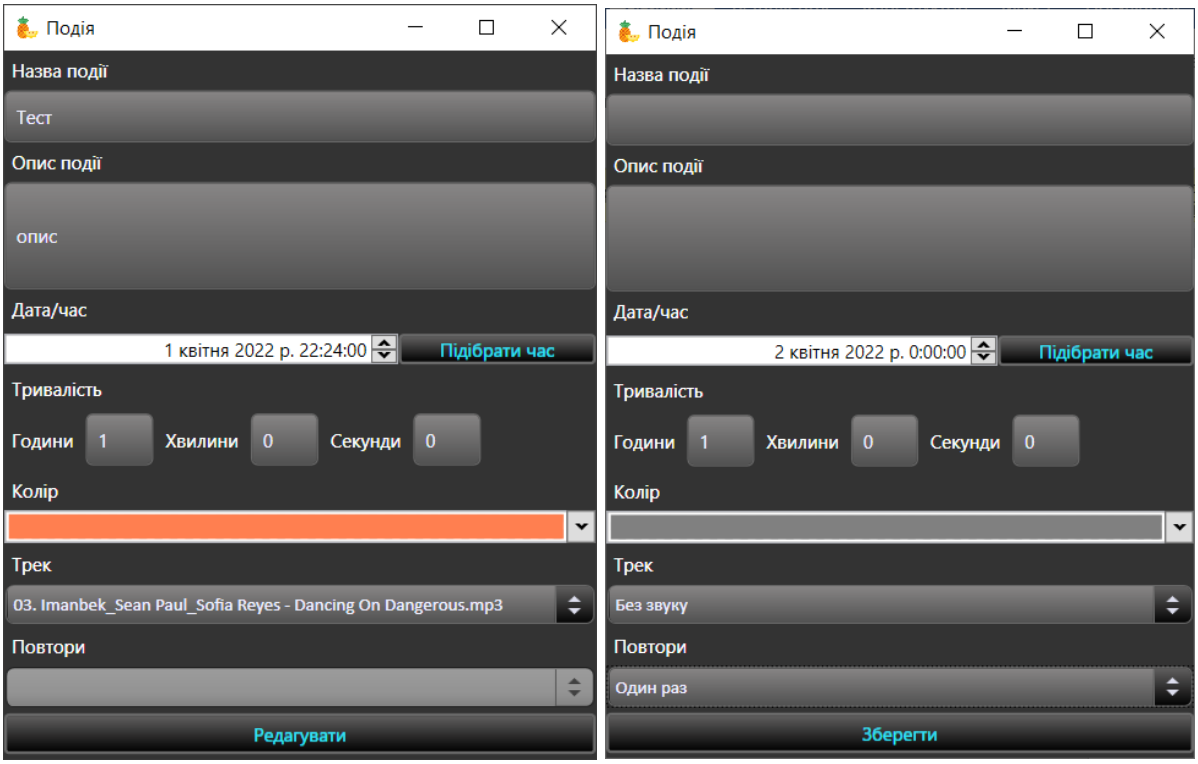

Рисунок 2.7 — Форма події

На даній формі розташовано заголовок та опис події, є поле для зручного вибору дати та часу, тривалість, колір зарисовки, та можливість обрати трек чи тишу як повідомлення про подію. Також є можливість повторювати подію 10 разів щодня, щотижня, щомісяця чи щороку. Також є можливість підібрати час. Для цього потрібно клікнути на однойменну кнопку.

З’явиться форма підбору дати (рис 2.8.). потрібно вказати дату початку (відкриється дата обрана на формі події) та обрати час який потрібно та період коли це можливо. Наприклад вам потрібно знайти 60 хвилин у робочий час у свому розкладі (наприклад для зустрічі з другом чи ділового ланчу). Для цього ви вказуєте обсяг часу рівним 60, а дату вказуєте поточною. Час від та до обираєте робочі години, тобто з 9:00 до 18:00 і програма сама перебере всі дати та підшукає перше вікно в заданому проміжку часу яке триватиме годину чи довше і виведе його початок у вигляді текстового повідомлення.

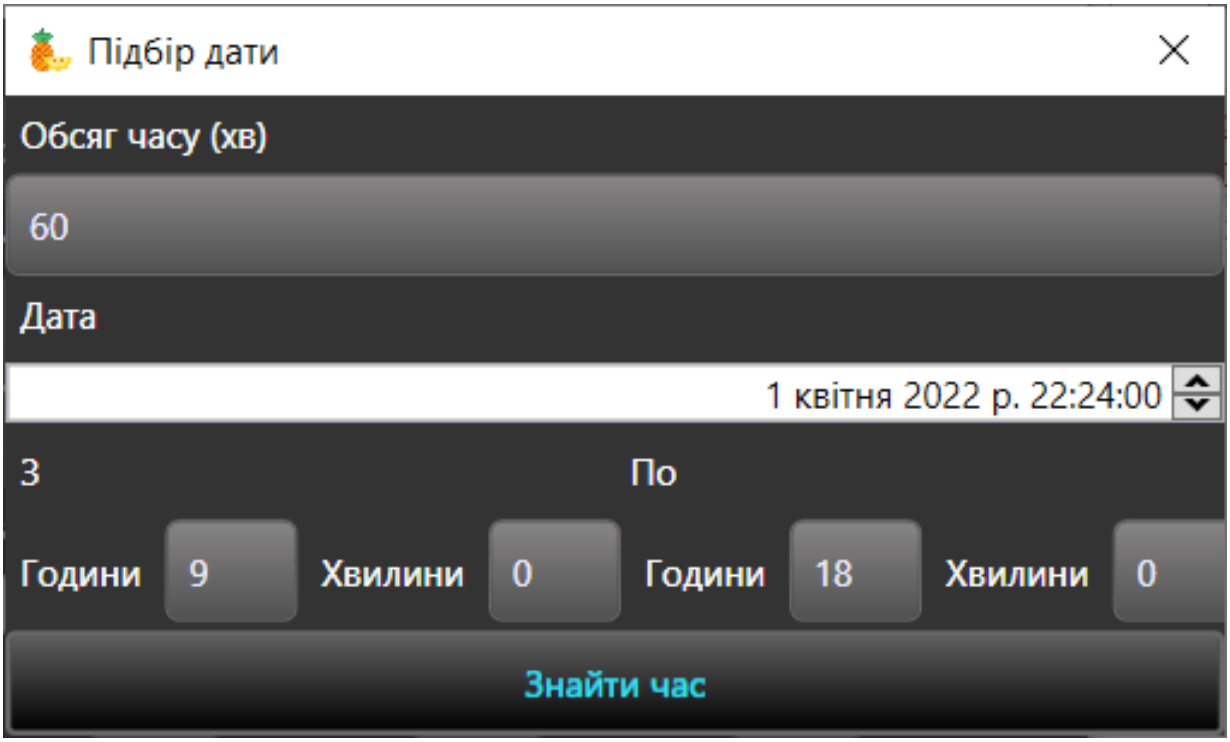

Рисунок 2.8 — Форма підбору дати

За потреби є можливість зберегти свій розклад на день у *.pdf файл. На жаль процес збереження доволі швидкий не вдається як слід зберегти зміну статусів збереження, проте вони є і доволі інформативні.

На рис. 2.9. показано зразок розкладу у форматі pdf. Як бачимо даний файл містить заголовок в якому міститься інфорація про дату та таблицю з інфомрацією про заголовок події, її опис час початку та загальну тривалість події.

Даний формат це конвертований ексель файл і він містить як переваги ексель так і його недоліки.

Серед переваг варто відмітити, що можна вказати формат тексту в шаблоні і всі наступні звіти будуть зроблені з вказаними шрифтами, відступами та інше.

Серед недоліків можна вказати те, що надто довгий текст не розширює комірку, а обрізається. Тобто він відсічеться і ніде окрім самої програми його не можна буде переглянути.

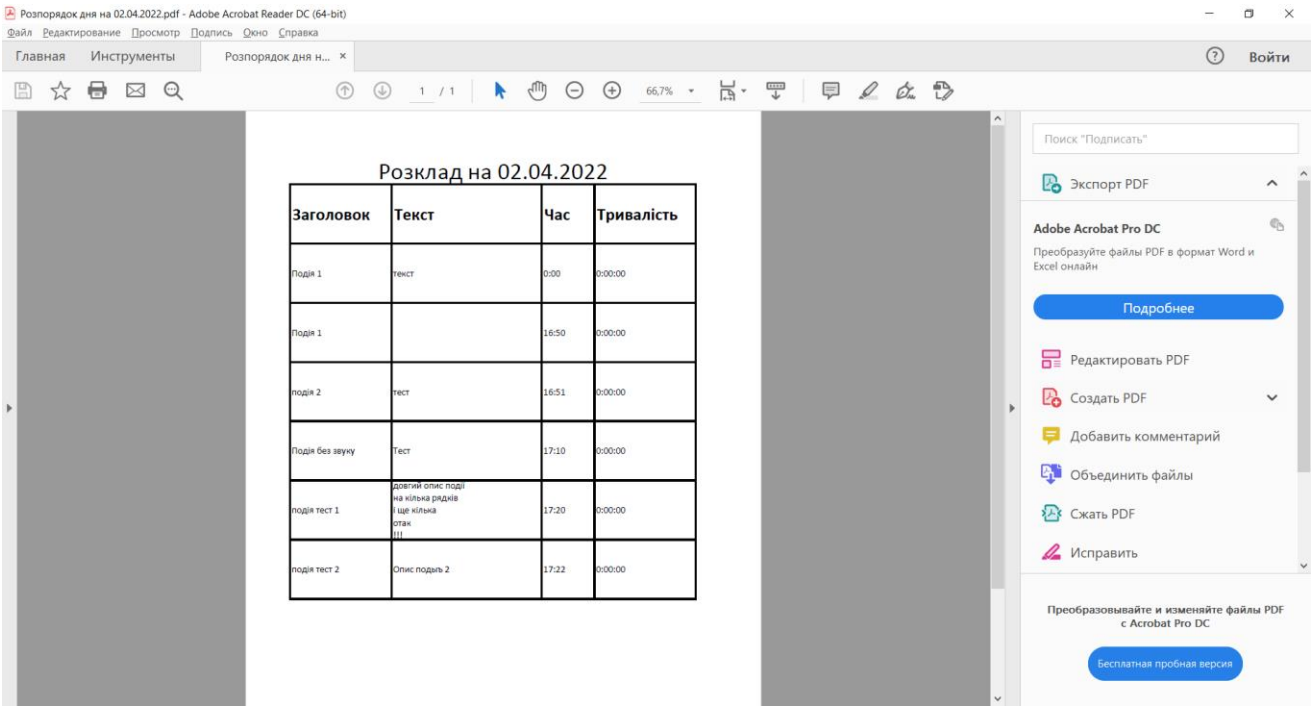

Рисунок 2.9 — Розклад на день

2.2 Проектування класів.

При розробці даного проекту було спроектовано 7 класів.

Розглянемо їх детальніше.

DataBase — клас описує внутрішній стан системи, в нього є дані про події, налаштування та аудіо, а також конструктор по замовчуванню, який створює пусту базу та узгоджує взаємозв’язки між сутностями при її створенні.

Core — даний клас описує основну роботу з даними. Він відповідає за серіалізацію та десеріалізацію бази, додавання нових події та аудіо, а також зберігає пароль під час сесії.

Settings — налаштування, зберігає дані про налаштування.

Event — подія, містить всю необхідну інформацію про подію.

HelpEvent — допоміжний клас для виведення події на форму.

Audio — аудіо, містить всю необхідну інформацію про аудіо.

ExcelReporter — статичний клас, містить функцію створення пдф звіту про розклад на день та функцію для звільнення ресурсів та процесів після роботи. Може бути розширений додатковими функціями для генерації звітів у форматі

ексель та csv та іншими форматами. Архітектура системи доволі гнучка і дозволяє доповнювати як існуючі сутності так і добавляти нові сутності без суттєвих змін у коді проекту.

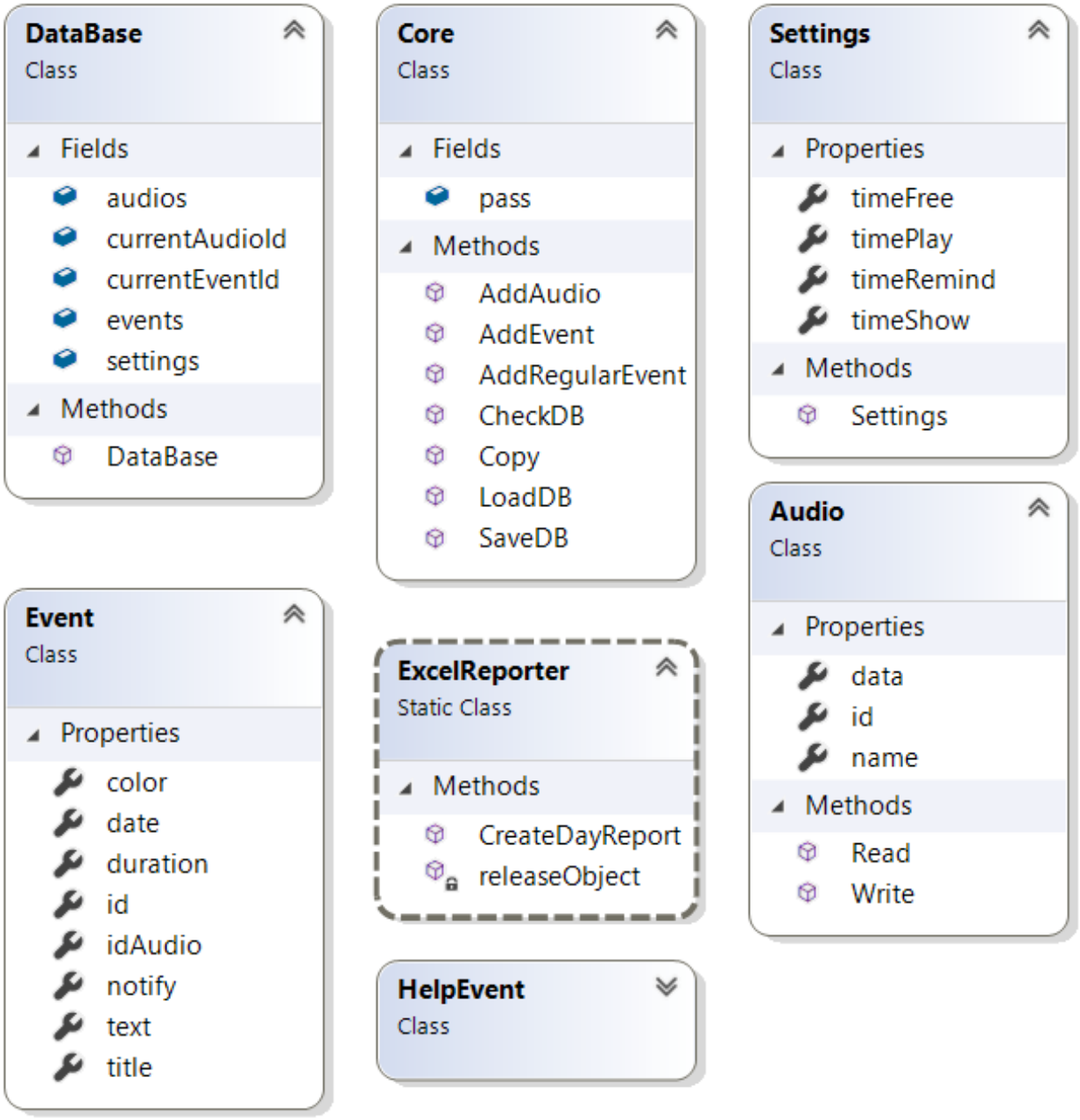

Рисунок 2.10 — Класи проекту

Розглянемо детальніше клас ExcelReporter.

Даний клас працює з об’єктами що знаходяться в бібліотеці Microsoft.Interop.Office.Excel.

Дана бібліотека розповсюджується безкоштовно і є частиною MS Office починаючи з версії 2007. Версії 2003 та більш ранні працювали з іншими форматами даних, в тому числі і форматом ексель.

Особливої уваги варта функція CreateDayReport. Вона приймає як параметр список подій та вказівник на прогресбар, який знаходиться на формі. Прогресбар потрібен аби відображати статус створення звіту, в той час як список подій і є даними для звіту.

Спершу відбувається перевірка чи в списку подій є хоча б одна. Якщо він не пустий, то функція ініціалізує об’єкти процесу ексель, документу ексель та книги ексель. Якщо ж список подій пустий, то функція завершує свою роботу і не створює ніякого звіту, навіть пустого. Робота буде відбуватися лише з об’єктом книги ексель, проте два інші необхідні аби її створити.

Далі формується шаблон, який включає шапку з написами та форматування стовпчиків. Також на даному етапі можна змінювати шрифти, кольори заливок та інше, проте параметри по замовчуванню підходять.

Після цього в циклі по подіях записуються події кожна в новий рядок.

Після того як всі події записані відбувається фінальне форматування документа. Туди входить все форматування яке неможливо з тих чи інших причин виконати перед записом даних. В конкретно нашому випадку туди входить зміна переносу всіх комірок в яких вписаний текст.

Після цього йде етап збереження файлу. Спершу викликається діалогове вікно, аби користувач обрав шлях до файлу. Дане вікно вже має назву файлу по замовчуванню, хоч користувач і може її змінити. Також вже обрано формат файлу, хоч, для зручності, можна переключитися на відображення всіх файлів.

Саме збереження відбувається засобами Excel та автоматичною конвертацією в формат pdf. Саме тому для роботи даної програми необхідний встановлений ексел від 2007 чи новіший. Після цього файл зберігається на диску. А всі ресурси звільняються.

Як побічний ефект процеси Excel можуть висіти ще в диспетчері задач, проте вони одразу ж зникнуть при закритті програми і ніяк не впливатимуть на роботу з Excel. Це особливість застарілої бібліотеки Microsoft та нових версій віндовс.

Було використано ще два класи не описані вище, оскільки вони розповсюджуються вільно в мережі інтернет.

Перший клас відповідає за шифрування та дешифрування даних.

Другий клас відповідає за зручний доступ до UI, а точніше до контролів DataGrid, які відображають таблиці.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

3.1 Реалізація програми.

При реалізації програми варто відмітити кілька цікавих моментів.

Найперше це використання стилю. Стиль складається з двох файлів кольорової схеми та власне ядра.

Частина кольорової схеми зоббражена на рис 3.1.

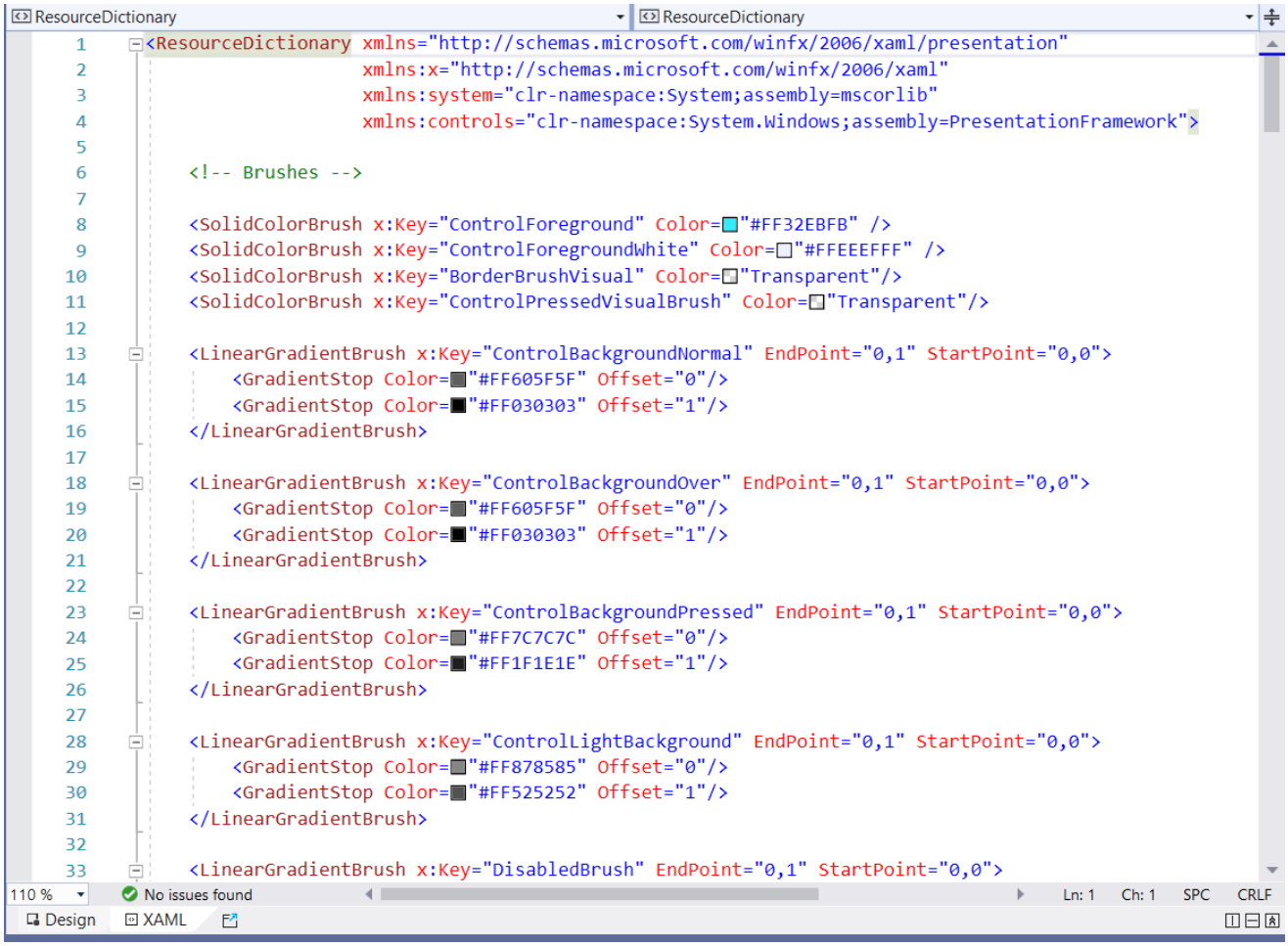

Рисунок 3.1 — Кольорова схема стилю

Ядро (логіка) стилю розміщена в іншому файлі та активно посилається на кольорову схему. Див рис 3.2.

Такий підхід дозволяє змінити кольори у всіх елементах стилю змінивши їх лише один раз.

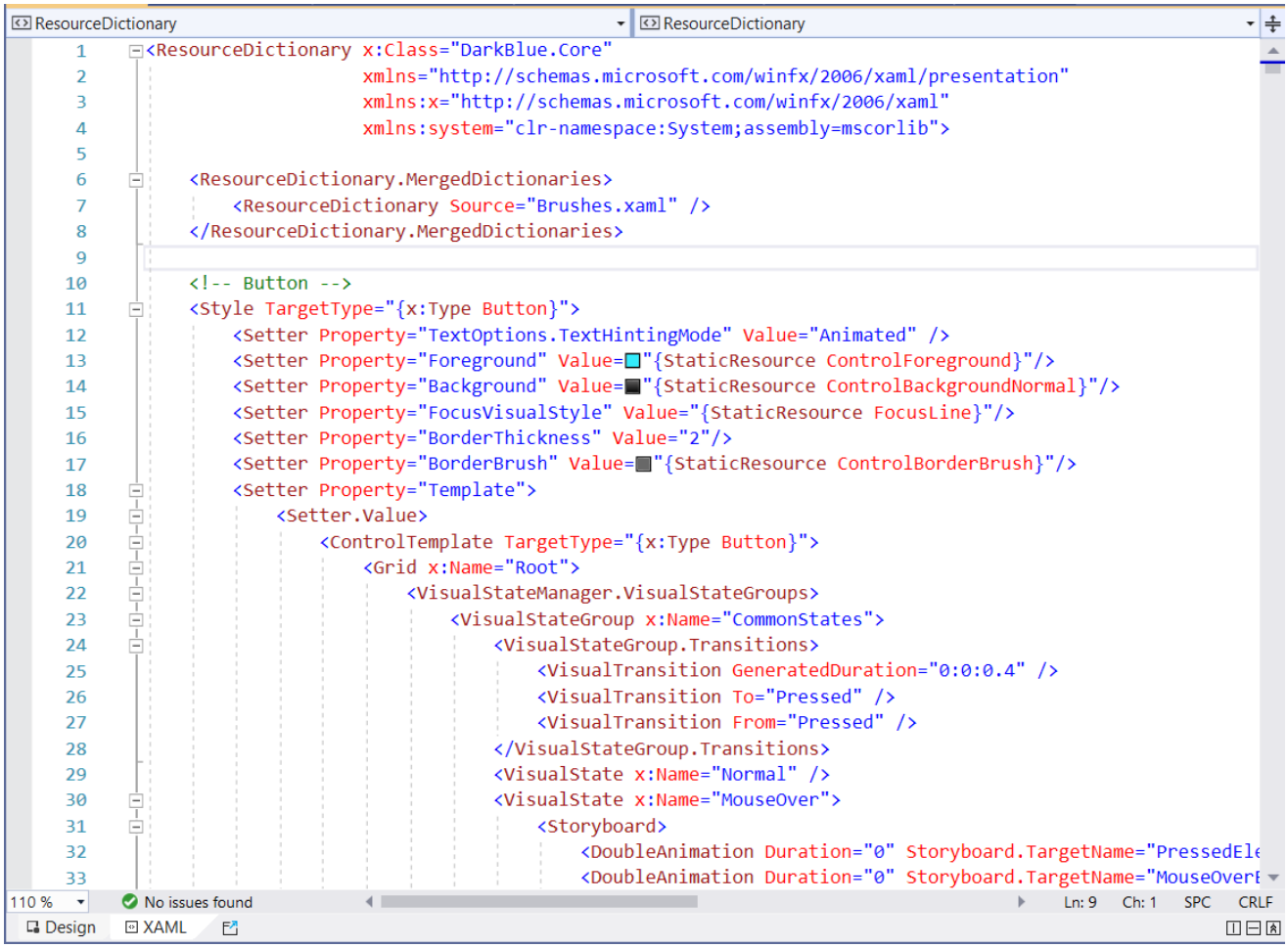

Рисунок 3.2 — Логіка стилю

Логіка одного елементу включає як анімацію при наведенні та зміні наведення так і триггери поведінки та шаблони форматів даних та самих користувацьких елементів. Така логіка є реалізованою для всіх класичних елементів управління.

Другим цікавим моментом є перетворення аудіо для збереження. Для цього його побайтово записано у відповідну змінну. Код для читання та відтворення показано на лістингах 3.1 та 3.2.

Лістинг 3.1. Читання аудіо

```
public void Read(string path)
{
   byte[] tmp = File.ReadAllBytes(path);
```

```
    data = Convert.ToBase64String(tmp); }
```

Лістинг 3.2. Запис аудіо

```
public void Write(string path)
{
   byte[] tmp = Convert.FromBase64String(data);
   File.WriteAllBytes(path, tmp);
}
```

Для відтворення аудіо використано віндовс плеєр. Для цього підключено бібліотеку WMPLib.

Також було підключено бібліотеку XCeed.WPF. в даній бібліотеці безліч різного корисного, проте в конкретно даному проекті цікаві лише розширені елементи керування. Було використано елементи DateTimeUpDown, який дозволяє зручно обирати дату та час, та ColorPicker, який дозволяє викликати діалог вибору кольору зі зразків чи кольорової діаграми. На жаль безкоштовна версія даної бібліотеки не дозволяє змінити стилі елементів на власні, тому вони можуть вибиватися з кольорової гами всієї програми.

Також варто згадати алгоритм відтворення сповіщень.

При запуску програми запускається таймер, котрий кожні 5 секунд перевіряє всі події. Якщо подія ще не відбулася, але повинна була відбутися, запускається її відтворення. При відтворенні звучить музика яку було обрано (або ж не звучить при відповідних налаштуваннях) та з’являється віндовс сповіщення з відповідним заголовком та текстом. Якщо таких подій декілька, то наступна виконається через 5 секунд на наступній ітерації таймера. Попередня музика змінюється новою, а сповіщення віндовс замінюється на нове, проте попереднє не видаляється, а поміщається в чергу віндовс сповіщень для подальшого перегляду (якщо саме сповіщення не було закрите). Для відтворення аудіо створюється тимчасовий

файл. Його одразу передається в плеєр, а сам файл видаляється. Музика звучить оскільки вона завантажена в плеєр (і, як наслідок, розміщена в оперативній пам’яті). Видалення файлу необхідне на випадок, якщо потрібно відвторити кілька сповіщень підряд або ж підійде черга чергового сповіщення поки не завершилося відтворення попереднього.

Показ сповіщення про подію показано на рис 3.3.

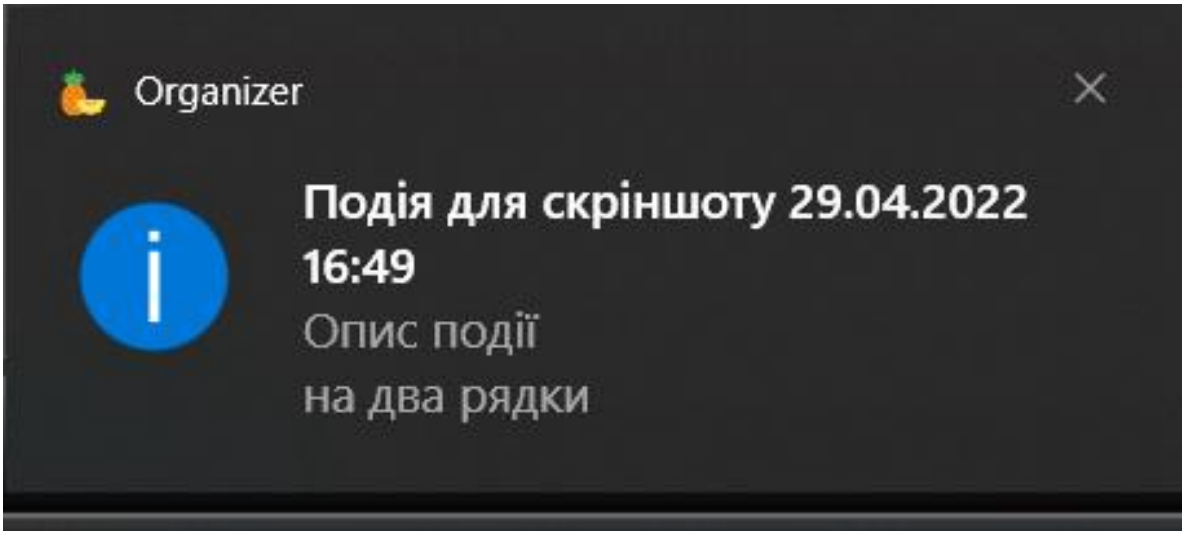

Рисунок 3.3 — Сповіщення про подію

Варто окремо виділити алгоритм створення звіту у форматі pdf.

Для початку викликається форма на якій розміщений лише прогресбар. У формі викликається метод створення звіту в новому потоці. Методу передається список подій в конкретний день та сам прогресбар. Це потрібно аби в подальшому показувати статус виконання.

В самій функції є безліч викликів інвоків для показу прогресу. Один з них показано на лістингу 3.3.

Лістинг 3.3. Інвоки

```
Application.Current.Dispatcher.Invoke(
    DispatcherPriority.Background,
    new ThreadStart(delegate
    {
```

Продовження лістингу 3.3.

```
            pb.IsIndeterminate = true;
            pb.Tag = "Створення файлу";
        }));
```

Як бачимо даний код викликає у новому потоці безіменний делегат (вказівник на функцію) яка змінює тип прогресбару (аби він запустився) та тег прогресбару. Тег прогресбару прив'язаний (binding) до напису на прогресбарі.

Сама структура показана на лістингу 3.4.

Лістинг 3.4. Форма прогресу

```
            xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
            xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
            xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
            xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
            xmlns:local="clr-namespace:Organizer"
            mc:Ignorable="d"
            Background="Transparent" AllowsTransparency="True" Height="50"
MinHeight="50" MaxHeight="50" Width="800" WindowStyle="None"
ResizeMode="NoResize" Loaded="Window_Loaded">
        <Grid>
            <ProgressBar Minimum="0" Maximum="100" Name="Status"  />
            <TextBlock Name="Textblock" Text="{Binding ElementName=Status,
Path=Tag}" HorizontalAlignment="Center" VerticalAlignment="Center" />
        </Grid></Window>
```

Як бачимо напис розташований по центру форми, а так як прогресбар займає всю форму, то і по центру прогресбару.

Параметри Background="Transparent", AllowsTransparency="True" та WindowStyle="None" дозволяють створити форму без рамки на якій розташований прогресбар та напис.

Тепер детальніше про саму генерацію звіту.

Для початку ми створюємо змінні під наш звіт (див лістинг 3.5.)

Лістинг 3.5. Ініціалізація змінних

```
Excel.Application xlApp;
Excel.Workbook xlWorkBook;
Excel.Worksheet xlWorkSheet;
object misValue = System.Reflection.Missing.Value;


xlApp = new Excel.Application();
xlWorkBook = xlApp.Workbooks.Add(misValue);
xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);
```

Після цього йде форматування, створення шапки та власне цикл в якому перебираються всі події та записуються у файл.

Після цього відбувається збереження ексель файлу у формат pdf. Проте перед цим, необхідно перевірити чи існує файл з такою ж назвою і якщо так видалити його. Детальніше на лістингу 3.6.

Лістинг 3.6. Збереження

```
if (File.Exists(savefile.FileName)) File.Delete(savefile.FileName);
            xlWorkBook.ExportAsFixedFormat(paramExportFormat,
               savefile.FileName, paramExportQuality,
               paramIncludeDocProps, paramIgnorePrintAreas, paramFromPage,
               paramToPage, paramOpenAfterPublish,
               misValue);
            xlWorkBook.Close(false, misValue, misValue);          xlApp.Quit();
```

Після збереження необхідно звільнити ресурси, інакше процес екселю залишиться активним, займатиме оперативну пам'ять і не дозволить адекватно працювати з іншими файлами. Для цього див лістинги 3.7 та 3.8

Лістинг 3.7. Звільнення ресурсів

```
releaseObject(xlWorkSheet);
releaseObject(xlWorkBook);
releaseObject(xlApp);
```

Лістинг 3.8. Функція звільнення ресурсів

```
private static void releaseObject(object obj)
{
   try
   {
      System.Runtime.InteropServices.Marshal.ReleaseComObject(obj);
      obj = null;
   }
   catch (Exception ex)
   {
      obj = null;
   }
   finally
   {
GC.Collect();
   }
}
```

Проте навіть при такому підході процес може залишитись активним після створення звіту. Проте він зникне після закриття програми і не мішатиме роботі з файлами.

3.2 Тестування програми.

Як показано на рис 3.4. Quality Control є лише одним із елементів Тестування Програмного Забезпечення.

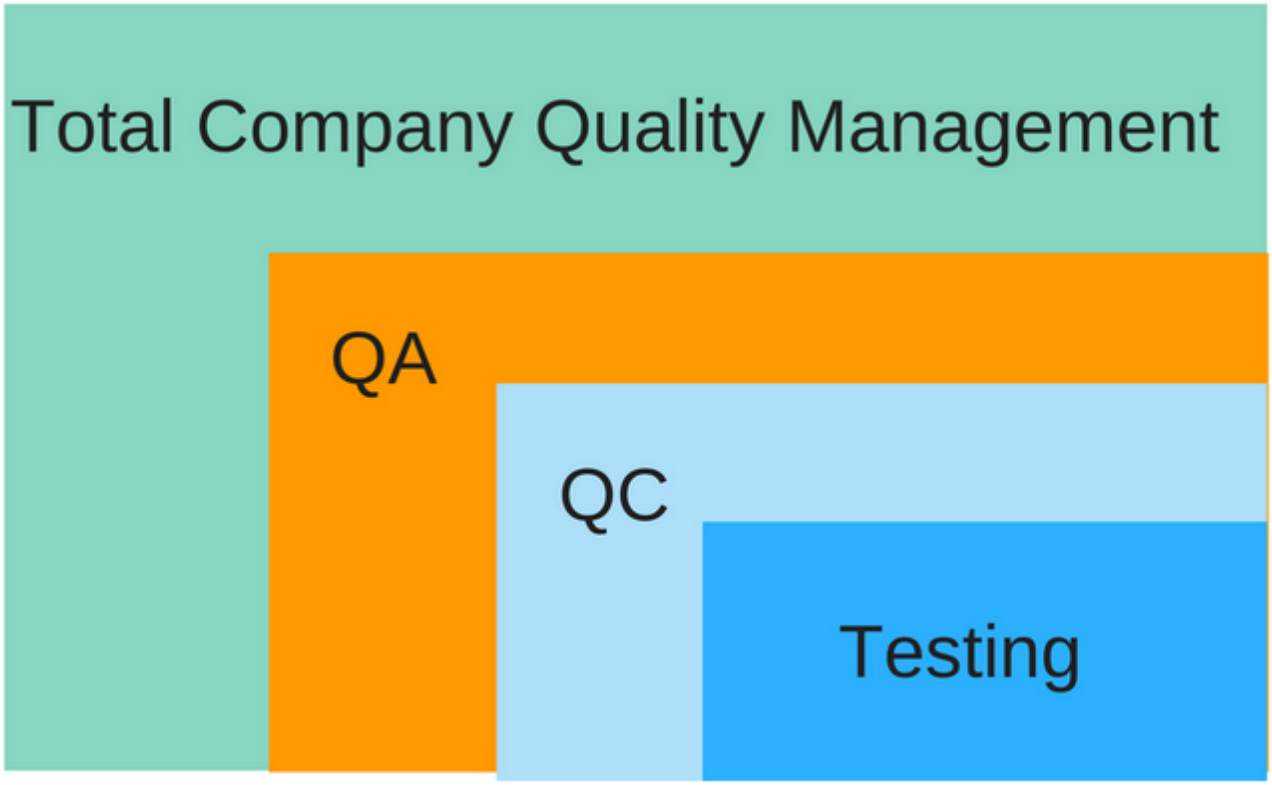

Рисунок 3.4 — Роль тестування у забезпеченні якості

І також неправильно розглядати Тестування Програмного Забезпечення тільки в контексті Верифікації та Валідації.

Верифікація — процес оцінки системи і її компонентів з метою співставлення результатів поточного етапу розробки, початково сформованим умовам. Тобто, виконання заданих завдань, цілей, строків по розробці продукту.

Валідація — це визначення відповідності ПЗ, що розробляється, очікуванням і потребам користувача, а також вимогам до системи.

Тестоване програмне забезпечення повинно проходити кожен з етапів тестування, обумовлених менеджментом компанії розробника та тест-дизайнерами для того, щоб вважати, що програмний продукт відносно якісний та придатний до використання.

Головними методами тестування програм є: тестування білої скриньки, тестування чорної скриньки та тестування сірої коробки.

Окрім того, ПЗ може бути протестоване в цілому, в компонентах / одиницях або ж у живій системі.

Однак на сьогодні розрізняють близько 150 підвидів тестування програмного забезпечення, щоб за необхідності протестувати програмний продукт від А до Я на усі 100%.

Ось кілька причин чому ж тестування в рамках даного процесу є дуже важливим:

— тестування дозволяє перевірити, чи правильно реалізовано вимоги до ПЗ, що розроблялось;

— тестування допомагає у виявленні дефектів / багів та забезпечує їх розпізнавання / вирішення до етапу розгортання програмного забезпечення;

— тестування пом’якшує наслідки та ризики втрат якщо програмний продукт все ж випустили по невірних вимогах. Вимоги в такому випадку намагаються частково виправити, переоцінити, а саму програму покращити;

— тестування також демонструє, що створене ПЗ відповідає вимогам до продуктивності;

— тестування допомагає перевірити належну інтеграцію та взаємодію програми з оточенням.

Люди схильні помилятися, людські помилки можуть призводити до порушення нормальної роботи програми на всіх етапах розробки програмного забезпечення, причому наслідки цього можуть бути дуже різноманітними — від незначних до катастрофічних.Наприклад форма на сайті не підсвічує, до тестування системи контролю запуску протонного колайдера або тестування безпечності використання рентгенівського апарату, який потенційно може спричинити шкоду людському здоров’ю. Для програмного забезпечення будь-якої складності неможливо врахувати всі проблеми. Проте тестування допомагає

виявити більшість помилок, з якими може зіткнутися користувач, й потенційно зменшує ризик виникнення проблем з ПЗ в подальшому на етапі використання.

Процес тестування в дечому подібний на детективне розслідування. Процес Тестування Програмного Забезпечення не завжди може бути передбачуваним, а кінцевий результат Тестування ПЗ цілком відомим.

Але робота тестера, полягає в розкритті таємної інформації на усьому шляху, яка допоможе людям приймати правильні рішення в майбутньому. Це набагато більше, ніж лише опрацювати специфікацію і шляхом порівняння зробити свою оцінку. Документації властивий фактор очікування.

Тестувальнику під час роботи потрібно мислити критично, задавати складні питання, зважувати ризики, помічати такі речі, які на перший погляд здаються неважливими, однак котрі при суворій експертизі набагато важливіші ніж здавалося й потребують подальшого дослідження.

Тестування програм не повинно сприйматися виключно як завдання пройти тест-кейси складені по опрацюванних вимогах. Звісно вимог необхідно дотримуватися, але варто іноді виходити за їх рамки.

Наскільки б вимоги не видавалися повними, вони ніколи не стануть вичерпним переліком, який відповість на питання як себе поведе ПЗ в реальних умовах. Завжди будуть вимоги, які не зазначені, які не передбачили або опустили. Звідси тест кейси не можуть повністю покрити програму.

На ділі перевірки та тестування системи повинні поєднуватися з розслідуванням та дослідженням. Дослідницьке тестування передбачає одночасне вивчення, тест-дизайн, виконання тестів. Тестер вивчає програму, знаходить нову інформацію, вивчає систему й у цьому процесі знаходить нові речі які б варто було покрити тестами. Наприклад, ми пройшли поріг acceptance criteria, але ми не можемо гарантувати якість програмного продукту і передавати його замовнику, тому що у ході дослідницького тестування були виявлені критичні баги, відбувся краш програми.

Тестування є вагомим на всіх етапах життєвого циклу розробки програмного забезпечення, а не тільки на етапі кодування. І найбільша частина цінності зосереджується на особі, яка його проводить — інтелекті тестувальника. Який допомагає виявляти проблеми вчасно, тобто до того, як вони наробили клопоту чи вилилися у витрати. Коли тестер робить вірне припущення, знову ж таки, інколи виходячи за рамки тестових випадків і скриптів та щось виявляє. Автоматизоване тестування та ПК ще не скоро на цій ланці замінять людський інтелект та смикалку.

Робота тестувальника — це постійне змагання, інколи потрібно придумувати нові способи, щоб щось протестувати або зрозуміти як воно повинне працювати. Уміння експериментувати та знаходити найкращі інструменти для роботи.

Через отаке творче начало іноді роботу тестувальника складно оцінити. Те ж саме також стосується оцінки якості програмного забезпечення. Робити оцінку за допомогою показників — оманливо. Хоча багатьом метрики подобаються: підрахунок кількості знайдених багів, кількість написаних і виконаних тестів, оцінка покриття програми тестами.

Левова частка роботи тестера — це спілкування та обмін інформацією. Тестери спілкуються та надають інформацію про якість програмного продукту, людям різних ролей, з різним рівнем підпорядкування по вертикалі управління, з різним рівнем знань. Щоб ці люди на основі інформації, яку їм надали тестери змогли прийняти правильні рішення. Розробники, власники продуктів, тестувальники, користувачі / клієнти, менеджери.

Ідеальний варіант, якщо в перспективі кожен тестувальник однаково удосконалюватиме як свої технічні навички, так і вміння спілкуватися з іншими людьми. Тестувальник зобов’язаний чітко висловлюватися, для цього вживати правильні слова та формулювання, щоб не бути неоднозначними. Обґрунтовано висловлювати припущення на основі фактів. Своєю поведінкою уникати непорозумінь. Не бути конфліктним.

Письмове спілкування настільки ж важливе. Це суттєва частина роботи тестувальника. Обирати спосіб спілкування для проекту потрібно враховуючи суб'єктивні чинники. Якщо виникають проблеми потрібно покращувати саме комунікацію в команді.

Тестування часто розглядається як щось, що може зробити будь-хто. Це правда в деякій мірі, кожен може вивчити продукт, задавати питання про нього, практично кожен здатен пройти крок за кроком тест-кейс або перевірити їх, відповідно до переліку вимог. Але аби робити це добре потрібні певні навики та особисті якості якими володіє далеко не кожен.

Тестування — це структурована технічна діяльність, це робота, котра дається не легко, якщо людина не має певних особистих якостей. потрібно навчитися з часом щось автоматизувати, а це в принципі не просте завдання. Це розуміння рамок автоматизації, визначення моментів, коли слід автоматизовувати, а коли краще перевірити вручну, знання коду, знання роботи API, розуміння таких інструментів, як Selenium, та багато іншого.

Нижче описано 5 етапів хорошого життєвого циклу тестування.

Етап перший. Аналіз вимог. Умови і критерії роботи системи, як правило, визначаються клієнтом або менеджером проєкту в процесі спілкування з клієнтом, або аналізом стандартів та нормативної документації. Це буде список функціональних та нефункціональних вимог.

Тестувальник працює над статичним тестуванням вимог: досліджує їх повноту, логічність, однозначність та визначеність. Також знаходить слабкі місця в тестовому покритті і виявляє потенцчійні ризики. Виникають обставини, коли наявні вимоги не тестуються, але використовуються на етапі дизайну і розробки. І вже готовий продукт направляється на контроль якості. Тоді кількість знайдених дефектів, вартість та масштаб їх виправлення можуть суттєво зрости та збільшити вартість розробки ледь не в кілька разів.

Етап другий. Процес дизайну.Один з найважливіших складових етапів, тому що користувачеві доведеться постійно взаємодіяти з системою. Продукт повинен

бути інтуїтивно зрозумілий для користувача, мати зрозумілий інтерфейс, а також бути оптимізованим аби користувач робив по мінімуму операцій для досягнення своїх цілей

На даному етапі тестувальник перевіряє існуючі прототипи ПЗ на відповідність вимогам замовника, коректності відображення візуальних елементів та зручність використання.

Етап третій. Розробка. Під час процесу розробки системи необхідно провести модульне, інтеграційне та системне тестування.

Для початку тестуються окремі компоненти програми кожен окремо. Це дозволяє виявити, в якій саме частині коду є помилка і досить швидко та дешево усунути її. Модульне тестування допомагає відстежити, чи не призвела зміна коду до появи нових помилок в уже перевірених місцях продукту. Також це допомагає краще зрозуміти роль кожного модуля системи. Усі знайдені дефекти, як правило, виправляються в коді без формального їх опису (внесення звітів в баг-трекер).

Інтеграційне тестування дозволяє перевірити, як компоненти коду об'єднуються і взаємодіють один з одним. Також воно визначає зв'язки між модулями, обладнанням або різними підсистемами та системами.

І системне тестування. На цій стадії мається на увазі перевірка всіх компонентів і модулів в цілому. Це необхідно для зменшення ймовірності дефектів, пов'язаних з особливостями поведінки системи в різних середовищах. В ході цих перевірок виявляються такі типи помилок, як: нераціональне використання ресурсів системи, несумісність з оточенням, збій або неправильне функціонування функціоналу та інші.

Етап четвертий. Процес тестування та відладки. Дебаггінг (відкладка) – це процес, під час якого знаходять і усувають помилки програми.

На даному етапі тестувальники перевіряють систему на наявність дефектів незалежно від того, чи відбувалось це раніше. Проводиться повне тестування інтерфейсу та функцій продукту. Всі виявлені помилки повинні бути задокументовані в баг-трекері. Також слід провести регресійне тестування, щоб

дослідити систему на предмет дефектів, які могли з'явитися після усунення інших помилок. Адже, як відомо, поганий програміст при усуненні одного багу створює три нові.

По завершенню процесу відкладки потрібно надати оцінку якості продукту, наскільки відповідають вимоги замовника реальній роботі системи.

Етап п'ятий. Експлуатація та підтримка. Після того, як продукт надходить у реліз, залишається необхідність у тестуванні, так як буде відбуватися оновлення програми чи її компонентів, будуть з'являтися нові баги, які були випущені або помилки, пов'язані з експлуатацією кінцевого користувача. В такому випадку потрібне втручання відділу контролю якості.

У разі виявлення користувачами тих чи інших пост-релізних багів, інформація про них передається у вигляді звітів про помилки команді розробки, яка, в залежності від серйозності проблеми, або негайно випускає виправлення (т.зв. hot-fix), або відкладає його до наступної версії програми. Є випадки коли зупинялися сервери онлайн ігор через те, що було виявлено експлойти, котрі дозволяють отримати доступ до ПК користувачів. Цей простій коштував мільйони доларів. При грамотному тестуванні дані експлойти можна було виявити раніше і здешевити витрати.

Всі зміни, що вносяться до програмного забезпечення, необхідно ретельно протестувати. ПЗ повинне продовжувати виконувати початкові бізнес-функції і не порушувати працездатність інших функцій і всієї системи в цілому.

Процес тестування ПЗ охоплює всі етапи життєвого циклу розробки. Він безперервний, тривалий і вимагає наявності досить досвідченої команди тестувальників, щоб охопити всі етапи тестування. А моніторинг та аналіз всього процесу допомагає спланувати і, та за необхідності, внести правки для підвищення ефективності подальших завдань. Ця частина сучасного процесу розробки ПЗ допомагає замовнику, команді розробників, і головне, кінцевим користувачам отримати продукт найвишчої якості.

Тестування програми це один з найважливіших процесів. Без нього неможливо створити жоден адекватний бізнес застосунок. Чим більша програма тим більше тестів потрібно для забезпечення належної якості. У медичній та деяких інших галузях є більш строгі вимоги до тестування, ніж у інших. Наприклад при розробці медичного обладнання повинні бути дотримані правила MISRA та покриття коду тестами від 97%. Даний показник може змінюватись в залежності від замовника, проте він завжди доволі жорсткий до виконання.

В таблиці 3.1. показано основні тесткейси та їх проходження.

Таблиця 3.1. Результати тестування.

№	Назва	Хід тестування	Очікуваний результат тестування	Результат тестування	Пройдено / непройдено
ТК1	Створення події	1. Запустити програму, ввести пароль. 2. Двічі клікнути на сьогоднішню дату. 3. Ввести назву у відповідне поле. 4. Натиснути кнопку “Добавити подію”. 5. Ввести поточний час та опис. 6. Натиснути кнопку “Зберегти”.	Відтворення події	Відтворення події	Успіх

Продовження таблиці 3.1.

ТК2	Створення події	1. Повторити ТК1 пункти 1-4. 2. Ввести час через 3 години. 3. Натиснути кнопку “Зберегти”.	Подія відобразиться в таблиці подій	Подія відобразиться в таблиці подій	Успіх
ТК3	Зміна налаштувань	1. Запустити програму, ввести пароль. 2. Натиснути на кнопку “Налаштування”. 3. Змінити налаштування. 4. Натиснути кнопку “Зберегти”. 5. Натиснути на кнопку “Налаштування”.	Налаштування змінено	Налаштування змінено	Успіх
ТК4	Зміна налаштувань	1. Повторити ТК3 пункти 1-3. 2. Закрити форму натиснувши хрестик 3. Натиснути на кнопку “Налаштування”.	Налаштування залишились старі	Налаштування залишились старі	Успіх

ВИСНОВКИ

В даній дипломній роботі було розроблено безпечний органайзер для навчання. Проаналізовано історію мов програмування та виділено популярні на даний час мови. Для реалізації проекту обрано мову програмування C#, оскільки вона найкраще підходить для даної задачі.

Оглянуто середовища розробки для C# та обрано MS Visual Studio 2019, через її націленість на дану мову та найкращий інструментарій для розробки.

Проаналізовано основних конкурентів.

Розроблено графічний інтерфейс користувача. Описано всі 9 графічних форм. Сам інтерфейс реалізований з допомогою технології WPF.

Також описано використані класи. Виділено їх особливості та призначення. Описано використані допоміжні класи, а також статичний клас для роботи зі звітами.

Спроектовано алгоритми роботи, виділено особливості проектування форми для відображення статусу та саму роботу зі створення pdf звіту. Розроблено стиль проекту та його складові частини. Виділено переваги такого підходу. Оглянуто підключені бібліотеки та обґрунтовано їх використання. Також в деталях оглянуто процес відтворення та перетворення аудіо файлів.

Протестовано програму та виправлено всі виявлені помилки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Галисеев Г. В. Ассемблер для Win 32. Самоучитель. — М.: Диалектика, 2007. — 368 с. — ISBN 978-5-8459-1197-1.

2. Зубков С. В. Ассемблер для DOS, Windows и UNIX. — М. ДМК Пресс; СПб. Питер, 2006. — 608 с. — ISBN 5-94074-259-9.

3. Роберт В Себеста. Основные концепции языков программирования = Concepts of Programming Languages. — 5-е изд. — М.: «Вильямс», 2001. — С. 672. — ISBN 5-8459-0192-8.

4. Языки программирования Ада, Си, Паскаль = Comparing and Assessong Programming Languages Ada, C, and Pascal / А.Фьюэр, Н.Джехани. — М.: Радио и Связь, 1989. — 368 с. — 50 000 экз. — ISBN 5-256-00309-7.

5. Прата С. Язык программирования С: Лекции и упражнения = C Primer Plus. — М.: Вильямс, 2006. — С. 960. — ISBN 5-8459-0986-4.

6. Джошуа Блох. Java. Эффективное программирование = Effective Java. — 3-е. — М.: Диалектика, 2019. — 464 с. — ISBN 978-5-6041394-4-8.

7. Язык программирования C# 5.0 и платформа .NET 4.5 Эндрю Троелсен

8. Meyer, Bertrand (1997). Object-Oriented Software Construction. Prentice Hall. ISBN 0-13-629155-4.

9. Гради Буч, Роберт А. Максимчук, Майкл У. Энгл, Бобби Дж. Янг, Джим Коналлен, Келли А. Хьюстон. Объектно-ориентированный анализ и проектирование с примерами приложений. М.: Вильямс, 2008.- 720с.

10. Гофман В.Э., Хомоненко А.Д. Delphi 6.– М., 2002.– 1145 с.

11. Культин Н.Б. Delphi 6. Программирование на Object Pascal.– М., 2002.– 526 с.

12. Електронні таблиці Microsoft Excel: Методичні вказівки до лабораторних робіт / Укл.: В.С. Сікора, І.В. Юрченко.– Чернівці: Рута, 2002.– 48 с.

13. Основи інформатики: Методичні вказівки до лабораторних робіт: У 2 ч./ Укл.: І.В. Юрченко.– Чернівці: Рута, 2000.– 79 с.

14. Система управління базами даних Microsoft Access: Методичні вказівки до лабораторних робіт / Укл.: В.С. Сікора, І.В. Юрченко.– Чернівці: Рута, 2002.– 40 с.

15. Комп'ютерні мережі: Методичні вказівки до лабораторних робіт / Укл.: В.С. Сікора, І.В. Юрченко.– Чернівці: Рута, 2002.– 43 с.

16. Операційна система Microsoft Windows: Методичні вказівки до лабораторних робіт / Укл.: В.С. Сікора, І.В. Юрченко.– Чернівці: Рута, 2003.– 48 с.

17. Текстовий редактор Microsoft Word: Методичні вказівки до лабораторних робіт / Укл.: В.С. Сікора, І.В. Юрченко.– Чернівці: Рута, 2003.– 56 с.

18. Семчук А.Р., Юрченко І.В. Економічна інформатика. Навчальний посібник.– Чернівці: МВІЦ "Місто", 2008.– 426 с.

19. Симонович С., Евсеев Г. Практическая информатика: универсальный курс.– М.: АСТ–ПРЕСС; Инфорком–Пресс, 1999.– 480 с.

20. Руденко В.Д. та ін. Базовий курс інформатики; за заг. ред. В.Ю.Бикова: [Навч. посіб.]. – К.: Вид. група BHV. – Кн. 1: Основи інформатики. – 2005. – 320 с.: іл.

21. Руденко В.Д. та ін. Базовий курс інформатики; за заг. ред. В.Ю.Бикова: [Навч. посіб.]. – К.: Вид. група BHV. – Кн. 2: Інформаційні технології. – 2006. – 368 с.: іл.

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
проф., д.т.н.. Азаров О.Д.

" " 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання бакалаврської дипломної роботи
"Безпечний органайзер з аудіосповіщенням для навчання"
08-23.БДР.006.00.003.ТЗ

Науковий керівник: доцент к.т.н.
_____ Савицька Л.А.

Студент групи 1КІ-18б
_____ Вторук О.В.

ВНТУ 2022

1 Підстава для виконання бакалаврської дипломної роботи (БДР)

1.1 Важливим є актуальність дослідження у напрямку дипломної роботи, яка обумовлена тим, що в даний час все більше студентів турбуються про те аби вчасно дістатися на пари та не забути про інші свої справи. При цьому коло завдань, що потребують вирішення, безперервно розширюється. Поряд з такими «базовими» завданнями, як нагадування про подію, все більший інтерес викликають і більш складні – безпека даних та кастомізація сповіщень;

1.2 Наказ про затвердження теми БДР.

2 Мета БДР і призначення розробки

2.1 Мета роботи — є розробка безпечного органайзера з аудіосповіщенням для навчання в середовищі Visual Studio 2019, який дозволить не забувати про пари та екзамени;

2.2 Призначення розробки — безпечний органаайзер зі аудіосповіщеннями для начання.

3 Вихідні дані для виконання БДР

3.1 Проведення аналізу існуючих органайзерів;

3.2 Розробка структури органайзера;

3.3 На основі структурних та функціональних схем здійснено розрахунок та моделювання елементів органайзера.

4 Вимоги до виконання БДР

Головна вимога — реалізувати програмну систему безпечного органайзера з аудіосповіщеннями.

5 Етапи БДР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 — Етапи БДР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Огляд і аналіз існуючих методів та рішень проектування органайзерів	09.02.22	16.02.22	**Виконав**
2	Дослідження методів розробки віконних додатків	17.02.22	24.02.22	**Виконав**
3	Проектування органайзера	29.02.22	21.03.22	**Виконав**
4	Апробація та впровадження результатів дослідження	22.03.22	20.04.22	**Виконав**
5	Опублікування результатів досліджень	21.04.22	15.05.22	**Виконав**
6	Оформлення пояснювальної записки, графічного матеріалу і презентації	16.05.22	05.06.22	**Виконав**
7	Підготовка супроводжуючих документів, їх підписування, проходження нормоконтролю та тесту на плагіат	16.06.22	16.06.22	**Виконав**

6 Матеріали, що подаються до захисту БДР

До захисту подаються: пояснювальна записка БДР, графічні і ілюстративні матеріали, протокол попереднього захисту БДР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до БДР українською та іноземною мовами, довідка про відповідність оформлення БДР діючим вимогам.

7 Порядок контролю виконання та захисту БДР

Виконання етапів графічної та розрахункової документації БДР контролюється науковим керівником згідно зі встановленими термінами. Захист БДР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання БДР

8.1 При оформлюванні БДР використовуються:

— ДСТУ 3008 : 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302 : 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104-2006 «Єдина система конструкторської документації. Основні написи»;

— Методичні вказівки. Кафедра обчислювальної техніки 2022;

— документами на які посилаються у вище вказаних.

8.2 Порядок виконання БДР викладено в «Положення про кваліфікаційні роботи на другому (бакалаврську) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21».

ДОДАТОК Б

Код програми

```
using Cipher;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Organizer
{
    public class Core
    {
        public static string pass = "123";
        public DataBase LoadDB(string pass)
        {
            string path = Environment.CurrentDirectory + "\\data";
            string json = "";
            string[] SB;
            try
            {
                SB = File.ReadAllLines(path);
                DataBase tmpDB = new DataBase();
                for (int i = 0; i < SB.Length; i++)
                    json += SB[i];
                string decryptJson = StringCipher.Decrypt(json, pass);
                DataBase db = JsonConvert.DeserializeObject<DataBase>(decryptJson);
```

```
        return db;
    }
    catch (Exception ex)
    {
        DataBase db = new DataBase() ;
        return db;
    }
}
public bool CheckDB(string pass)
{
    string path = Environment.CurrentDirectory + "\\data";
    string json = "";
    string[] SB;
    try
    {
        SB = File.ReadAllLines(path);
        DataBase tmpDB = new DataBase();
        for (int i = 0; i < SB.Length; i++)
            json += SB[i];
        string decryptJson = StringCipher.Decrypt(json, pass);

        return true;
    }
    catch (Exception ex)
    {
        return false;
    }
}
```

```
public void SaveDB(DataBase db, string pass)
{
    string path = Environment.CurrentDirectory + "\\data";
    string json = JsonConvert.SerializeObject(db, Formatting.Indented);
    string encryptJson = StringCipher.Encrypt(json, pass);
    File.WriteAllText(path, encryptJson);

}
public void AddEvent(Event e, DataBase db)
{
    e.id = db.currentEventId++;
    e.notify = true;
    db.events.Add(e);
}
public void AddAudio(Audio a, DataBase db)
{
    a.id = db.currentAudioId++;
    db.audios.Add(a);
}
public void AddRegularEvent(Event e, DataBase db, int type)
{
    AddEvent(e, db);

    if (type == 0)
        return;

    Event tmp;
    DateTime tmpdt = e.date;
    int count = 10;
```

```
    for (int i = 0; i < count; i++)
    {
        if (type == 1)
            tmpdt = tmpdt.AddDays(1);
        else if (type == 2)
            tmpdt = tmpdt.AddDays(7);
        else if (type == 3)
            tmpdt = tmpdt.AddMonths(1);
        else if (type == 4)
            tmpdt = tmpdt.AddYears(1);
        else
        {
            ; // сюди не повинно заходити
        }

        tmp = Copy(e);
        tmp.date = tmpdt;
        AddEvent(tmp, db);
    }
}
public Event Copy(Event e)
{
    Event copy = new Event();
    copy.title = e.title;
    copy.text = e.text;
    copy.color = e.color;
    return copy;
}
```

```
    }
}
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Media;

namespace Organizer
{
    public class DataBase
    {
        public int currentEventId;
        public List<Event> events;
        public int currentAudioId;
        public List<Audio> audios;
        public Settings settings;
        public DataBase()
        {
            events = new List<Event>();
            audios = new List<Audio>();
            settings = new Settings();
        }
    }
    public class Event
    {
        public int id { get; set; }
```

```
    public string title { get; set; }
    public string text { get; set; }
    public DateTime date { get; set; }
    public TimeSpan duration { get; set; }
    public int idAudio { get; set; }
    public Brush color { get; set; }
    public bool notify { get; set; }
}
public class HelpEvent
{
    public int id { get; set; }
    public string title { get; set; }
    public string text { get; set; }
    public string date { get; set; }
    public string duration { get; set; }
    public Brush color { get; set; }
    public HelpEvent(Event e)
    {
        id = e.id;
        title = e.title;
        text = e.text;
        date = e.date.ToShortTimeString();
        duration = e.duration.ToString();
        color = e.color;
    }
}
public class Audio
{
    public int id { get; set; }
```

```
    public string name { get; set; }
    public string data { get; set; }
    public void Read(string path)
    {
        byte[] tmp = File.ReadAllBytes(path);
        data = Convert.ToBase64String(tmp);
    }
    public void Write(string path)
    {
        byte[] tmp = Convert.FromBase64String(data);
        File.WriteAllBytes(path, tmp);
    }
}
public class Settings
{
    public int timeRemind { get; set; } //за ск хв до події нагадати
    public int timePlay { get; set; } //ск сек відтворювати аудіо
    public int timeShow { get; set; } //ск хв показувати сповіщення
    public int timeFree { get; set; } //ск хв між подіями

    public Settings()
    {
        timeRemind = 15;
        timePlay = 20;
        timeShow = 30;
        //timeFree = 10;
    }

}
```

```
}
using System.Windows.Controls;
using System.Windows.Controls.Primitives;
using System.Windows.Media;

namespace DataGridHelperNS
{
   static class DataGridHelper
   {
      static public DataGridCell GetCell(DataGrid dg, int row, int column)
      {
         DataGridRow rowContainer = GetRow(dg, row);

         if (rowContainer != null)
         {
            DataGridCellsPresenter presenter =
GetVisualChild<DataGridCellsPresenter>(rowContainer);

            // try to get the cell but it may possibly be virtualized
            DataGridCell cell =
(DataGridCell)presenter.ItemContainerGenerator.ContainerFromIndex(column);
            if (cell == null)
            {
               // now try to bring into view and retreive the cell
               dg.ScrollIntoView(rowContainer, dg.Columns[column]);
               cell =
(DataGridCell)presenter.ItemContainerGenerator.ContainerFromIndex(column);
            }
            return cell;
```

```
        }
        return null;
    }

    static public DataGridRow GetRow(DataGrid dg, int index)
    {
        DataGridRow row =
(DataGridRow)dg.ItemContainerGenerator.ContainerFromIndex(index);
        if (row == null)
        {
            // may be virtualized, bring into view and try again
            dg.ScrollIntoView(dg.Items[index]);
            row =
(DataGridRow)dg.ItemContainerGenerator.ContainerFromIndex(index);
        }
        return row;
    }

    static T GetVisualChild<T>(Visual parent) where T : Visual
    {
        T child = default(T);
        int numVisuals = VisualTreeHelper.GetChildrenCount(parent);
        for (int i = 0; i < numVisuals; i++)
        {
            Visual v = (Visual)VisualTreeHelper.GetChild(parent, i);
            child = v as T;
            if (child == null)
            {
                child = GetVisualChild<T>(v);
```

```
            }
            if (child != null)
            {
                break;
            }
        }
        return child;
    }
  }
}
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading;
using System.Windows;
using System.Windows.Forms;
using System.Windows.Threading;
using Microsoft.Win32;
using Application = System.Windows.Application;
using Excel = Microsoft.Office.Interop.Excel;

namespace Organizer
{
  public static class ExcelReporter
  {
    public static void CreateDayReport(List<Event> le,
System.Windows.Controls.ProgressBar pb)
    {
```

```
if (le.Count == 0)
    return;

Application.Current.Dispatcher.Invoke(
    DispatcherPriority.Background,
    new ThreadStart(delegate
    {
        pb.IsIndeterminate = true;
        pb.Tag = "Створення файлу";
    }));

Excel.Application xlApp;
Excel.Workbook xlWorkBook;
Excel.Worksheet xlWorkSheet;
object misValue = System.Reflection.Missing.Value;

xlApp = new Excel.Application();
xlWorkBook = xlApp.Workbooks.Add(misValue);
xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);

Application.Current.Dispatcher.Invoke(
    DispatcherPriority.Background,
    new ThreadStart(delegate
    {
        pb.IsIndeterminate = true;
        pb.Tag = "Створення шапки";
    }));

xlWorkSheet.get_Range("A4").ColumnWidth = 20;
```

```
xlWorkSheet.get_Range("B4").ColumnWidth = 30;
xlWorkSheet.get_Range("C4").ColumnWidth = 10;
xlWorkSheet.get_Range("D4").ColumnWidth = 20;

xlWorkSheet.get_Range("F1").RowHeight = 30;
xlWorkSheet.get_Range("F2").RowHeight = 10;

xlApp.DisplayAlerts = false;
var excelcells = xlWorkSheet.get_Range("A1", "D1");
excelcells.Merge(Type.Missing);
excelcells.HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;
excelcells.VerticalAlignment = Excel.XlVAlign.xlVAlignCenter;
excelcells.Font.Size = 28;

xlApp.DisplayAlerts = true;

xlWorkSheet.Cells[1, 1] = "Розклад на " + le[0].date.ToShortDateString();

xlWorkSheet.Cells[2, 1] = "Заголовок";
xlWorkSheet.Cells[2, 2] = "Текст";
xlWorkSheet.Cells[2, 3] = "Час";
xlWorkSheet.Cells[2, 4] = "Тривалість";

excelcells = xlWorkSheet.get_Range("A2", "D2");
excelcells.Font.Size = 20;
excelcells.Font.Bold = true;

int current = 2;
for (int i = 0; i < le.Count; i++)
```

```
        {
            current++;

            Application.Current.Dispatcher.Invoke(
                DispatcherPriority.Background,
                new ThreadStart(delegate
                {
                    pb.IsIndeterminate = true;
                    pb.Tag = "Запис події " + (i+1).ToString() + " з " + le.Count;

                }));

            xlWorkSheet.Cells[current, 1] = le[i].title;
            xlWorkSheet.Cells[current, 2] = le[i].text;
            xlWorkSheet.Cells[current, 3] = le[i].date.ToShortTimeString();
            xlWorkSheet.Cells[current, 4] = le[i].duration.ToString();
        }

        excelcells = xlWorkSheet.get_Range("A2", "D" + current);
        excelcells.WrapText = true;
        xlWorkSheet.get_Range("A2", "D" + current).Cells.Borders.Weight =
Excel.XlBorderWeight.xlMedium;
        xlWorkSheet.get_Range("A2", "D" + current).Cells.VerticalAlignment =
Excel.XlVAlign.xlVAlignCenter;
        xlWorkSheet.get_Range("A2", "D" + current).Cells.HorizontalAlignment =
Excel.XlHAlign.xlHAlignLeft;
        xlWorkSheet.get_Range("A2", "D" + current).Cells.RowHeight = 68;

        Application.Current.Dispatcher.Invoke(
```

```
    DispatcherPriority.Background,
    new ThreadStart(delegate
    {
        Microsoft.Win32.SaveFileDialog savefile = new
Microsoft.Win32.SaveFileDialog();
        savefile.FileName = "Розпорядок дня на " + le[0].date.ToShortDateString() +
".pdf";
        savefile.Filter = "PDF files (*.pdf)|*.pdf|All files (*.*)|*.*";

        Excel.XlFixedFormatType paramExportFormat =
Excel.XlFixedFormatType.xlTypePDF;
        Excel.XlFixedFormatQuality paramExportQuality =
            Excel.XlFixedFormatQuality.xlQualityStandard;
        bool paramOpenAfterPublish = false;
        bool paramIncludeDocProps = true;
        bool paramIgnorePrintAreas = true;
        object paramFromPage = 1;// Type.Missing;
        object paramToPage = 2;// Type.Missing;

        pb.IsIndeterminate = true;
        pb.Tag = "Збереження у файл";

        if (savefile.ShowDialog() == true)
        {
            if (File.Exists(savefile.FileName)) File.Delete(savefile.FileName);
            //xlWorkBook.SaveAs(savefile.FileName,
Excel.XlFileFormat.xlWorkbookNormal, misValue, misValue, misValue, misValue,
Excel.XlSaveAsAccessMode.xlExclusive, misValue, misValue, misValue, misValue,
misValue);
```

```
        xlWorkBook.ExportAsFixedFormat(paramExportFormat,
            savefile.FileName, paramExportQuality,
            paramIncludeDocProps, paramIgnorePrintAreas, paramFromPage,
            paramToPage, paramOpenAfterPublish,
            misValue);
        xlWorkBook.Close(false, misValue, misValue);
        xlApp.Quit();
    }

    releaseObject(xlWorkSheet);
    releaseObject(xlWorkBook);
    releaseObject(xlApp);
}));
    }

    private static void releaseObject(object obj)
    {
        try
        {
            System.Runtime.InteropServices.Marshal.ReleaseComObject(obj);
            obj = null;
        }
        catch (Exception ex)
        {
            obj = null;
        }
        finally
        {
            GC.Collect();
```

```
            }
        }
    }
}
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Threading;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Threading;
using System.Windows.Forms;
using System.Drawing;

namespace Organizer
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
```

```
public partial class MainWindow : Window
{
    public DataBase db;
    DispatcherTimer timer = new DispatcherTimer();
    DispatcherTimer stoptimer = new DispatcherTimer();
    WMPLib.WindowsMediaPlayer wplayer = new WMPLib.WindowsMediaPlayer();
    public NotifyIcon NI = new NotifyIcon();

    public MainWindow()
    {
        InitializeComponent();
    }

    private void MainCalendar_MouseDoubleClick(object sender, MouseButtonEventArgs e)
    {
        WindowEvents we = new WindowEvents();
        we.db = db;
        we.date =  (DateTime)MainCalendar.SelectedDate;
        we.Show();
    }

    private void Window_Loaded(object sender, RoutedEventArgs e)
    {
        if (File.Exists(Environment.CurrentDirectory + "\\data"))
        {
            WindowLogin wl = new WindowLogin();
            wl.ShowDialog();
```

```
            db = new Core().LoadDB(Core.pass);

            timer.Tick += new EventHandler(timer_Tick);
            timer.Interval = new TimeSpan(0, 0, 5);
            timer.Start();

            stoptimer.Tick += new EventHandler(stoptimer_Tick);
            stoptimer.Interval = new TimeSpan(0, 0, db.settings.timePlay);
        }
        else
        {
            db = new DataBase();

            new Core().SaveDB(db, "123");
        }
    }

    private void timer_Tick(object sender, EventArgs e)
    {
        for (int i = 0; i < db.events.Count; i++)
        {
            if(db.events[i].date.AddMinutes(-db.settings.timeRemind)  <DateTime.Now
&& db.events[i].notify)
            {
                try
                {
                    NI.BalloonTipText = db.events[i].text;
                    NI.BalloonTipTitle = db.events[i].title + " " +
db.events[i].date.ToString("g");
```

```
                NI.BalloonTipIcon = ToolTipIcon.Info;
                NI.Icon = new Icon(Environment.CurrentDirectory + "\\pineapple_fruit_food_icon_218369.ico");
                NI.Visible = true;
                NI.ShowBalloonTip(db.settings.timeShow * 60 * 1000);
            }
            catch { }


            wplayer.controls.stop();
            File.Delete(Environment.CurrentDirectory + "\\audio.mp3");
            db.events[i].notify = false;
            new Core().SaveDB(db, Core.pass);
            if (db.events[i].idAudio != 0)
            {
                try
                {
                    Audio audio;
                    if (db.events[i].idAudio > 0)
                    {
                        audio = db.audios.Where(a => a.id == db.events[i].idAudio).First();
                        audio.Write(Environment.CurrentDirectory + "\\audio.mp3");

                        wplayer.URL = Environment.CurrentDirectory + "\\audio.mp3";
                        wplayer.controls.play();
                        File.Delete(Environment.CurrentDirectory + "\\audio.mp3");
                    }
                    else
                    {
```

```
                    ;
                }
            }
            catch
            {
                continue;
            }

            stoptimer.Start();
//              wplayer.controls.stop();

        }
        break;
      }
    }
  }
  private void stoptimer_Tick(object sender, EventArgs e)
  {
    wplayer.controls.stop();
    stoptimer.Stop();
  }
  private void Button_Click(object sender, RoutedEventArgs e)
  {
    WindowAudios wa = new WindowAudios();
    wa.db = db;
    wa.ShowDialog();
  }

  private void Button_Click_1(object sender, RoutedEventArgs e)
```

```
        {
            WindowSettings ws = new WindowSettings();
            ws.db = db;
            ws.ShowDialog();
        }
    }
}
using Microsoft.Win32;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Forms;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using OpenFileDialog = Microsoft.Win32.OpenFileDialog;

namespace Organizer
{
    /// <summary>
```

```
    /// Interaction logic for WindowAudios.xaml
    /// </summary>
    public partial class WindowAudios : Window
    {
        WMPLib.WindowsMediaPlayer wplayer = new WMPLib.WindowsMediaPlayer();
        public NotifyIcon NI = new NotifyIcon();
        public DataBase db;
        public WindowAudios()
        {
            InitializeComponent();
        }

        void Play(object sender, RoutedEventArgs e)
        {
            try
            {
                wplayer.controls.stop();
                File.Delete(Environment.CurrentDirectory + "\\audio.mp3");
            }
            catch { }
            for (var vis = sender as Visual; vis != null; vis = VisualTreeHelper.GetParent(vis) as Visual)
            {
                if (vis is DataGridRow)
                {
                    try
                    {
```

```
db.audios[DGAudios.SelectedIndex].Write(Environment.CurrentDirectory +
"\\audio.mp3");
                    wplayer.URL = Environment.CurrentDirectory + "\\audio.mp3";
                    wplayer.controls.play();
                    File.Delete(Environment.CurrentDirectory + "\\audio.mp3");

                }
                catch { }
            }
        }
/*          NI.BalloonTipText = "Текст";
        NI.BalloonTipTitle = "Заголовок";
        NI.BalloonTipIcon = ToolTipIcon.Info;
        NI.Icon = new Icon(Environment.CurrentDirectory +
"\\pineapple_fruit_food_icon_218369.ico");
        NI.Visible = true;
        NI.ShowBalloonTip(1000);*/
        //          NI.ShowBalloonTip(5000, "Hi", "This is a BallonTip from Windows
Notification", ToolTipIcon.Info);
    }

    private void Window_Loaded(object sender, RoutedEventArgs e)
    {
        ToForm();
    }
    public void ToForm()
    {
        DGAudios.ItemsSource = null;
```

```
    DGAudios.ItemsSource = db.audios;
}
private void FRAdd_Click(object sender, RoutedEventArgs e)
{
    OpenFileDialog OFD = new OpenFileDialog();
    OFD.Filter = "mp3 files (*.mp3)|*.mp3| all files (*.*)|*.*";
    Title = "mp3 трек";
    if (OFD.ShowDialog() == true)
    {
        string path = OFD.FileName;
        Audio a = new Audio();
        a.Read(path);
        a.name = new FileInfo(path).Name;
        new Core().AddAudio(a, db);
        ToForm();
        new Core().SaveDB(db, Core.pass);
    }
}

private void FRDelete_Click(object sender, RoutedEventArgs e)
{
    try
    {
        db.audios.RemoveAt(DGAudios.SelectedIndex);
        ToForm();
        new Core().SaveDB(db, Core.pass);
    }
    catch { }
}
```

```
        private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
        {
            wplayer.controls.stop();
            File.Delete(Environment.CurrentDirectory + "\\audio.mp3");
        }


    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace Organizer
{
    /// <summary>
```

```
    /// Interaction logic for WindowChangePass.xaml
    /// </summary>
    public partial class WindowChangePass : Window
    {
        public WindowChangePass()
        {
            InitializeComponent();
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            if (new Core().CheckDB(OldPass.Password))
            {
                DataBase db = new Core().LoadDB(OldPass.Password);
                new Core().SaveDB(db, NewPass.Password);
                Close();
            }
            else
            {
                MessageBox.Show("Пароль не вірний", "Пароль", MessageBoxButton.OK,
MessageBoxImage.Warning);
            }
        }
    }
}
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace Organizer
{
    /// <summary>
    /// Interaction logic for WindowEvent.xaml
    /// </summary>
    public partial class WindowEvent : Window
    {
        public DateTime date;
        public bool isNew;
        public DataBase db;
        public Event ev;
        public ObservableCollection<String> audio = new
ObservableCollection<string>();
        public WindowEvent()
        {
            InitializeComponent();
        }
```

```
        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            try
            {
                audio.Add("Без звуку");
                for (int i = 0; i < db.audios.Count; i++)
                    audio.Add(db.audios[i].name);
                cmbAudio.ItemsSource = null;
                cmbAudio.ItemsSource = audio;
                cmbAudio.SelectedIndex = 0;
            }
            catch
            {
                MessageBox.Show("Завантажте аудіо", "Аудіо", MessageBoxButton.OK,
MessageBoxImage.Warning);
            }

            if (isNew)
            {
                Do.Content = "Зберегти";
                Color.SelectedColor = Colors.Gray;
                cmbRepeat.SelectedIndex = 0;
                Date.Value = date;
            }
            else
            {
                Do.Content = "Редагувати";
                Color.SelectedColor                                                =
((System.Windows.Media.SolidColorBrush)(ev.color)).Color;
```

```
            cmbRepeat.IsEnabled = false;
            Date.Value = ev.date;

            Hour.Text = ev.duration.Hours.ToString();
            Min.Text = ev.duration.Minutes.ToString();
            Sec.Text = ev.duration.Seconds.ToString();

            Title.Text = ev.title;
            Text.Text = ev.text;
            try
            {
                cmbAudio.SelectedIndex = db.audios.IndexOf(db.audios.Where(a => a.id
== ev.idAudio).First()) + 1;
            }
            catch
            {
                cmbAudio.SelectedIndex = 0;
//                      MessageBox.Show("Щось не так з списком аудіо", "Аудіо",
MessageBoxButton.OK, MessageBoxImage.Warning);
            }

        }
    }

    private void Do_Click(object sender, RoutedEventArgs e)
    {
        if (isNew)
        {
            ev = new Event();
```

```
            try
            {
                ev.title = Title.Text;
                ev.text = Text.Text;
                ev.date = (DateTime)Date.Value;
                ev.duration = new TimeSpan(Convert.ToInt32(Hour.Text),
Convert.ToInt32(Min.Text), Convert.ToInt32(Sec.Text));
                for (int i = 0; i < db.events.Count; i++)
                {
                    if ((db.events[i].date <= ev.date &&
                        db.events[i].date + db.events[i].duration >= ev.date) ||

                        (db.events[i].date <= ev.date + ev.duration &&
                        db.events[i].date + db.events[i].duration >= ev.date + ev.duration) ||

                        (db.events[i].date <= ev.date &&
                        db.events[i].date + db.events[i].duration >= ev.date+ev.duration) ||

                        (db.events[i].date >= ev.date &&
                        db.events[i].date + db.events[i].duration <= ev.date + ev.duration)
                        )
                    {
                        MessageBox.Show("Запланована подія " + db.events[i].title + " на "
+ db.events[i].date.ToShortTimeString() + " додавання нової події на цей час неможливе",
                            "Подія",
                            MessageBoxButton.OK,
                            MessageBoxImage.Warning);
                        return;
```

```
            }
        }

        if (db.audios.Count == 0 || cmbAudio.SelectedIndex == 0)
            ev.idAudio = -1;
        else
            ev.idAudio = db.audios[cmbAudio.SelectedIndex - 1].id;

        ev.color = new SolidColorBrush((Color)Color.SelectedColor);
        new Core().AddRegularEvent(ev,db,cmbRepeat.SelectedIndex);
        Close();
      }
      catch { }

    }
    else
    {
      try
      {
        ev.title = Title.Text;
        ev.text = Text.Text;
        ev.date = (DateTime)Date.Value;
        ev.duration = new TimeSpan(Convert.ToInt32(Hour.Text),
Convert.ToInt32(Min.Text), Convert.ToInt32(Sec.Text));
        for (int i = 0; i < db.events.Count; i++)
        {
          if (((db.events[i].date <= ev.date &&
            db.events[i].date + db.events[i].duration >= ev.date) ||
```

```
            (db.events[i].date <= ev.date + ev.duration &&
            db.events[i].date + db.events[i].duration >= ev.date + ev.duration) ||

            (db.events[i].date <= ev.date &&
            db.events[i].date + db.events[i].duration >= ev.date + ev.duration) ||

            (db.events[i].date >= ev.date &&
            db.events[i].date + db.events[i].duration <= ev.date + ev.duration)
            )
            &&
            (db.events[i] != ev)
            )
        {
            MessageBox.Show("Запланована подія " + db.events[i].title + " на "
+ db.events[i].date.ToShortTimeString() + " додавання нової події на цей час неможливе",
                "Подія",
                MessageBoxButton.OK,
                MessageBoxImage.Warning);
            return;
        }
    }

    if (db.audios.Count == 0 || cmbAudio.SelectedIndex == 0)
        ev.idAudio = -1;
    else
        ev.idAudio = db.audios[cmbAudio.SelectedIndex - 1].id;

    ev.color = new SolidColorBrush((Color)Color.SelectedColor);
```

```
                Close();
            }
            catch { }
        }
    }

    private void Button_Click(object sender, RoutedEventArgs e)
    {
        if (!isNew)
            date = ev.date;

        WindowFind wf = new WindowFind();
        wf.d = date;
        wf.db = db;
        wf.ShowDialog();
    }
  }
}
using DataGridHelperNS;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
```

```
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace Organizer
{
    /// <summary>
    /// Interaction logic for WindowEvents.xaml
    /// </summary>
    public partial class WindowEvents : Window
    {
        public DataBase db;
        public DateTime date;
        string now;
        public WindowEvents()
        {
            InitializeComponent();
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            WindowState = WindowState.Maximized;
            ToForm();
        }
        public void ToForm()
        {
/*          Event e1 = new Event();
            e1.color = Brushes.Red;
```

```
    e1.title = "будильник";
    e1.text = "мій будильник";
    e1.date = DateTime.Now;

    Event e2 = new Event();
    e2.color = Brushes.Yellow;
    e2.title = "будильник";
    e2.text = "мій будильник";
    e2.date = DateTime.Now.AddHours(1);

    db.events.Add(e1);
    db.events.Add(e2);*/
    List<HelpEvent> lhe = new List<HelpEvent>();
    now = date.ToShortDateString();
    for (int i = 0; i < db.events.Count; i++)
    {
        if (db.events[i].date.ToShortDateString() == now)
            lhe.Add(new HelpEvent(db.events[i]));
    }

    DGEvents.ItemsSource = null;
    DGEvents.ItemsSource = lhe;

}

private void FRAdd_Click(object sender, RoutedEventArgs e)
{
    WindowEvent we = new WindowEvent();
    we.db = db;
```

```
            we.isNew = true;
            we.date = date;
            we.ShowDialog();
            ToForm();
            new Core().SaveDB(db, Core.pass);
        }

        private void DGEvents_MouseDoubleClick(object sender,
MouseButtonEventArgs e)
        {
            try
            {
                WindowEvent we = new WindowEvent();
                we.db = db;
                we.ev = db.events.Where(sp => sp.id ==
Convert.ToInt32(DataGridHelper.GetCell(DGEvents, DGEvents.SelectedIndex,
0).ToString().Substring(38))).First();
                we.isNew = false;
                we.ShowDialog();
                ToForm();
                new Core().SaveDB(db, Core.pass);
            }
            catch { }
        }

        private void FRDelete_Click(object sender, RoutedEventArgs e)
        {
            try
            {
```

```
            WindowEvent we = new WindowEvent();
            we.db = db;
            Event ev = db.events.Where(sp => sp.id == Convert.ToInt32(DataGridHelper.GetCell(DGEvents, DGEvents.SelectedIndex, 0).ToString().Substring(38))).First();
            db.events.Remove(ev);
            ToForm();
            new Core().SaveDB(db, Core.pass);
        }
        catch { }
    }

    private void ToPDF_Click(object sender, RoutedEventArgs e)
    {
        now = date.ToShortDateString();
        WindowProgress wp = new WindowProgress();
        wp.db = db;
        wp.le = db.events.Where(ev => ev.date.ToShortDateString() == now).ToList();
        wp.mode = 1;
        wp.ShowDialog();
    }
  }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
```

```
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace Organizer
{
    /// <summary>
    /// Interaction logic for WindowFind.xaml
    /// </summary>
    public partial class WindowFind : Window
    {
        public DateTime d;
        public DataBase db;

        int hFrom;
        int mFrom;
        int hTo;
        int mTo;
        DateTime date;
        int mFind;
        DateTime tmp;
        public WindowFind()
        {
            InitializeComponent();
        }
```

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
   Date.Value = d;
}

private void Button_Click(object sender, RoutedEventArgs e)
{
   try
   {
      hFrom = Convert.ToInt32(HourFrom.Text);
      mFrom = Convert.ToInt32(HourTo.Text);
      hTo = Convert.ToInt32(HourTo.Text);
      mTo = Convert.ToInt32(MinTo.Text);
      date = (DateTime)Date.Value;
      mFind = Convert.ToInt32(Time.Text);
      List<Event> le = db.events.Where(ev => ev.date >= date || ev.date + ev.duration >= date).ToList();
      le = le.OrderBy(ev => ev.date).ToList();

      tmp = date;
      FixDate(tmp);
      for (int i = 0; i < le.Count; i++)
      {
         if ((le[i].date - tmp).TotalMinutes >= mFind) // якщо від tmp до найближчої події є mFind, то знайшли,
         {
            MessageBox.Show(tmp.ToString("g"), "Час", MessageBoxButton.OK, MessageBoxImage.Information);
```

```
                break;
            }
            else
            {
                tmp = le[i].date + le[i].duration;
                FixDate(tmp);
            }
        }
        Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Помилка", MessageBoxButton.OK,
MessageBoxImage.Error);
    }
}
public void FixDate(DateTime d)
{
    if (tmp.Hour <= hFrom && tmp.Minute <= mFrom) // поточна дата до
потрібного часу, встановити поточну дату на старт поточного часу
    {
        tmp = new DateTime(tmp.Year, tmp.Month, tmp.Day, hFrom, mFrom, 0);
    }
    if (tmp.Hour >= hTo && tmp.Minute >= mTo)// якщо поточна дата після
потрібного часу, встановити поточну дату на старт наступного дня
    {
        tmp = tmp.AddDays(1);
        tmp = new DateTime(tmp.Year, tmp.Month, tmp.Day, hFrom, mFrom, 0);
    }
```

```
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace Organizer
{
    /// <summary>
    /// Interaction logic for WindowLogin.xaml
    /// </summary>
    public partial class WindowLogin : Window
    {
        int count = 0;
        public bool isOk = false;
        public WindowLogin()
        {
            InitializeComponent();
```

```
    }

    private void Button_Click(object sender, RoutedEventArgs e)
    {
        if (new Core().CheckDB(Pass.Password))
        {
            isOk = true;
            Close();
        }
        else
        {
            count++;
            MessageBox.Show("Пароль не вірний", "Пароль", MessageBoxButton.OK,
MessageBoxImage.Warning);
            if(count>3)
                Application.Current.Shutdown();
        }
    }

    private void Button_Click_1(object sender, RoutedEventArgs e)
    {
        WindowChangePass wcp = new WindowChangePass();
        wcp.ShowDialog();
    }

    private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
    {
        if (!isOk)
```

```
            e.Cancel = true;
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using System.Windows.Threading;

namespace Organizer
{
    /// <summary>
    /// Interaction logic for WindowProgress.xaml
    /// </summary>
    public partial class WindowProgress : Window
    {
        public DataBase db;
        public int mode;
```

```
        public List<Event> le;

        public WindowProgress()
        {
            InitializeComponent();
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            WindowStartupLocation = WindowStartupLocation.CenterScreen;
            if (mode == 1)
                new Thread(() => Do()).Start();
            else
            {

            }
        }
        public void Do()
        {
            ExcelReporter.CreateDayReport(le, Status);
            Application.Current.Dispatcher.Invoke(
             DispatcherPriority.Background,
                new ThreadStart(delegate
                {
                    Close();
                }));
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace Organizer
{
    /// <summary>
    /// Interaction logic for WindowSettings.xaml
    /// </summary>
    public partial class WindowSettings : Window
    {
        public DataBase db;
        public WindowSettings()
        {
            InitializeComponent();
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
```

```
            TimePlay.Text = db.settings.timePlay.ToString();
            TimeShow.Text = db.settings.timeShow.ToString();
            TimeRemind.Text = db.settings.timeRemind.ToString();
//            TimeFree.Text = db.settings.timeFree.ToString();
        }

        private void Save_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                db.settings.timePlay = Convert.ToInt32(TimePlay.Text);
                db.settings.timeShow = Convert.ToInt32(TimeShow.Text);
                db.settings.timeRemind = Convert.ToInt32(TimeRemind.Text);
//                db.settings.timeFree = Convert.ToInt32(TimeFree.Text);
                Close();
            }
            catch { }
        }
    }
}
```

ДОДАТОК В

Слайди презентації

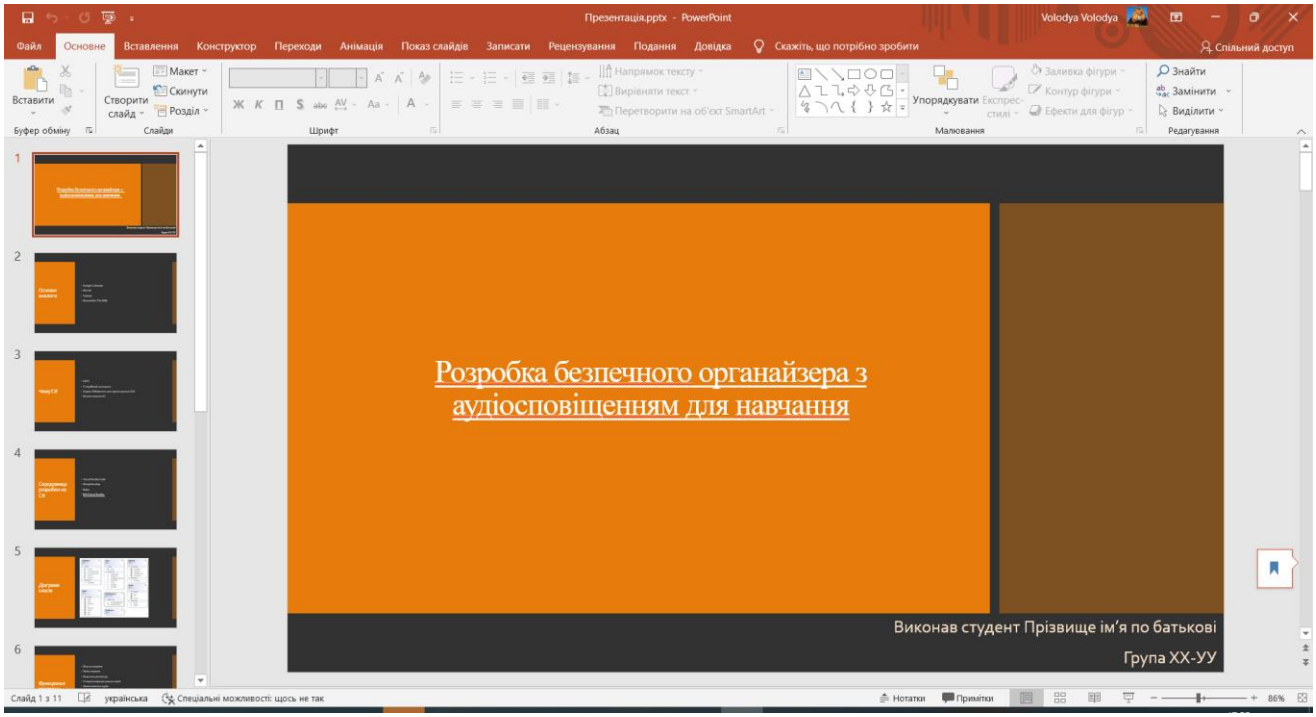

Рисунок В.1 — Тема роботи

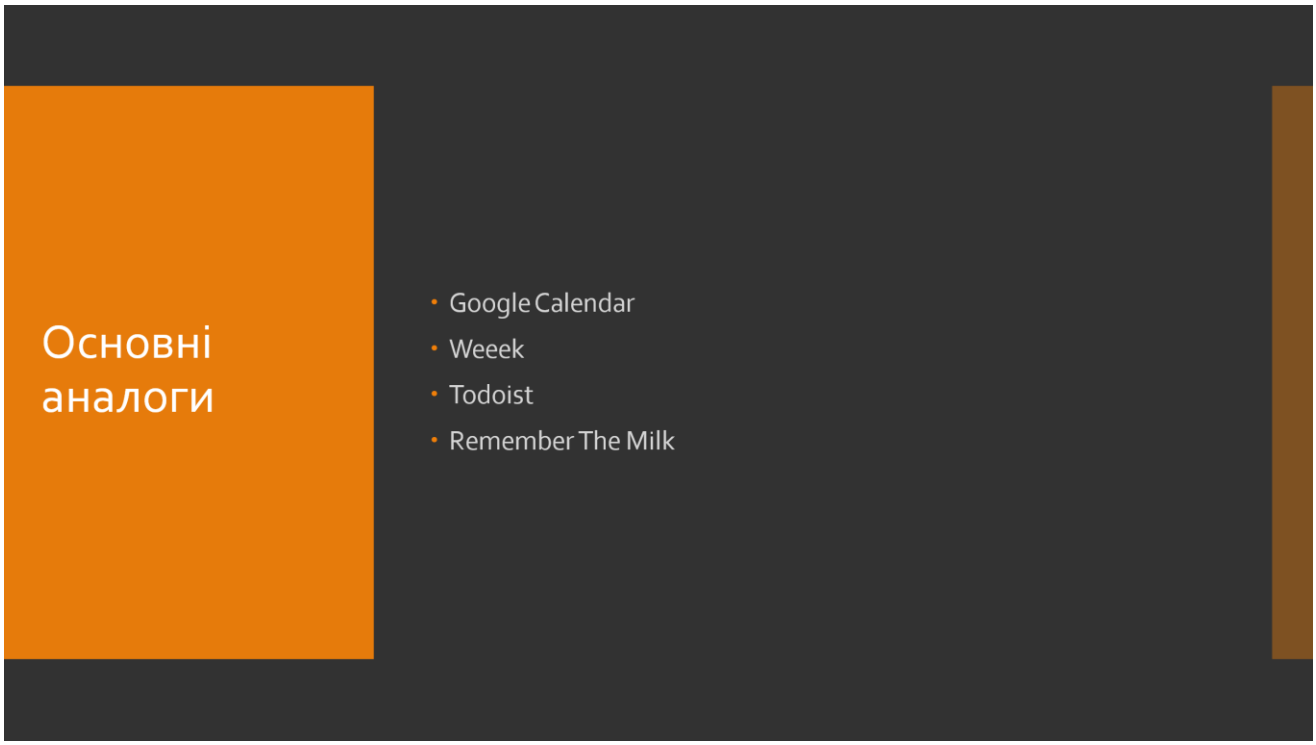

Рисунок В.2 — Основні аналоги

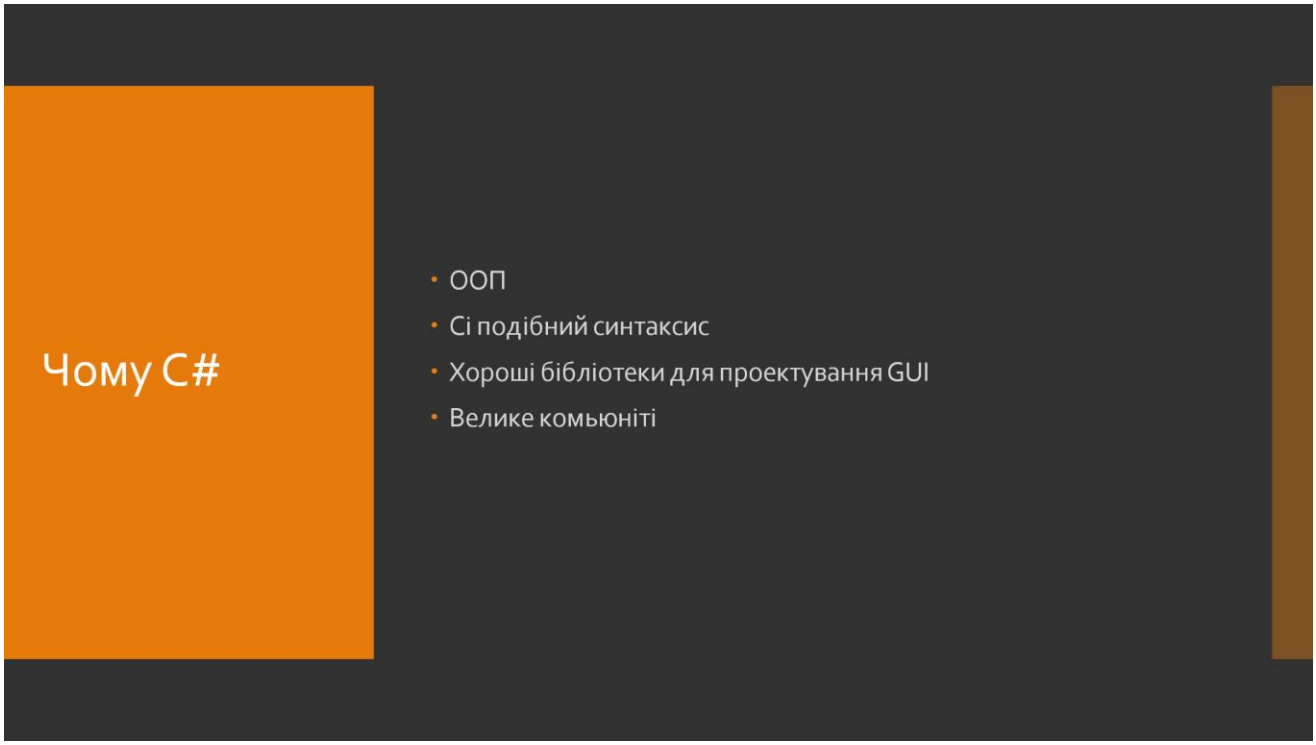

Рисунок В.3 — Причини вибору C#

ДОДАТОК Г

Початок презентації

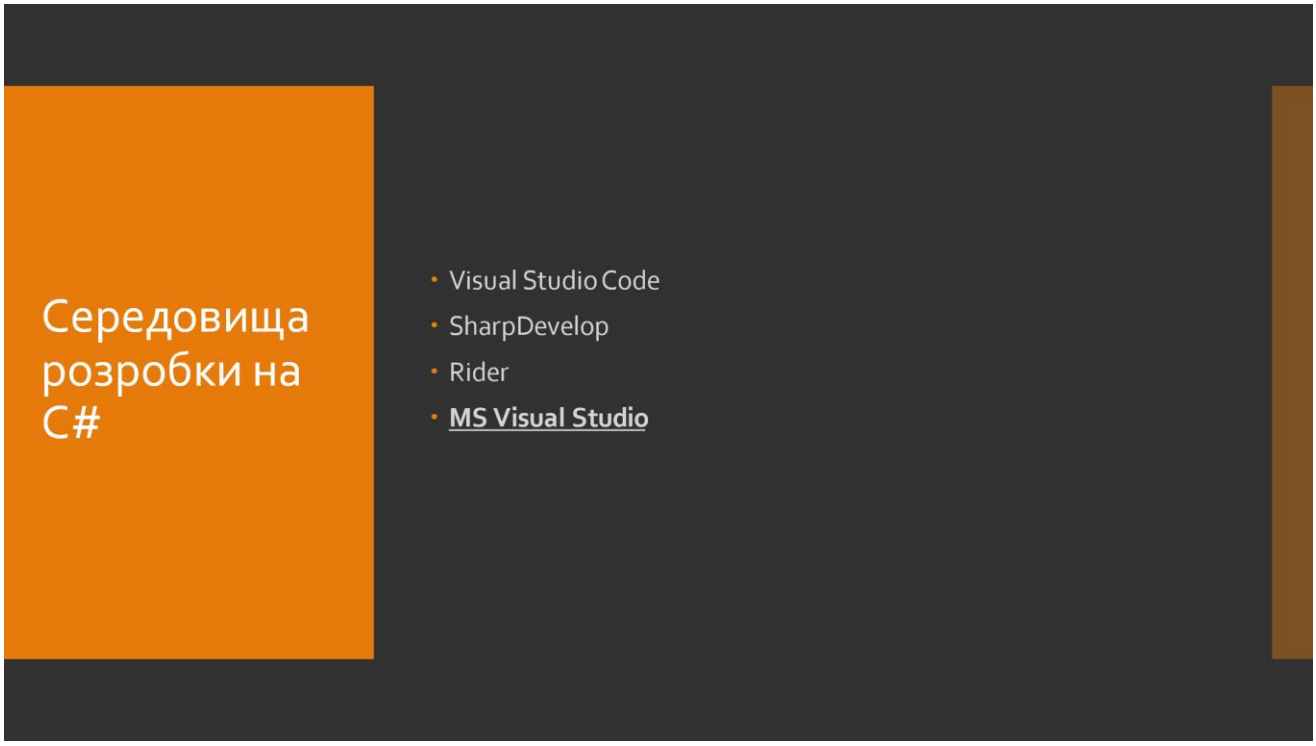

Рисунок Г.1 — Середовища розробки C#

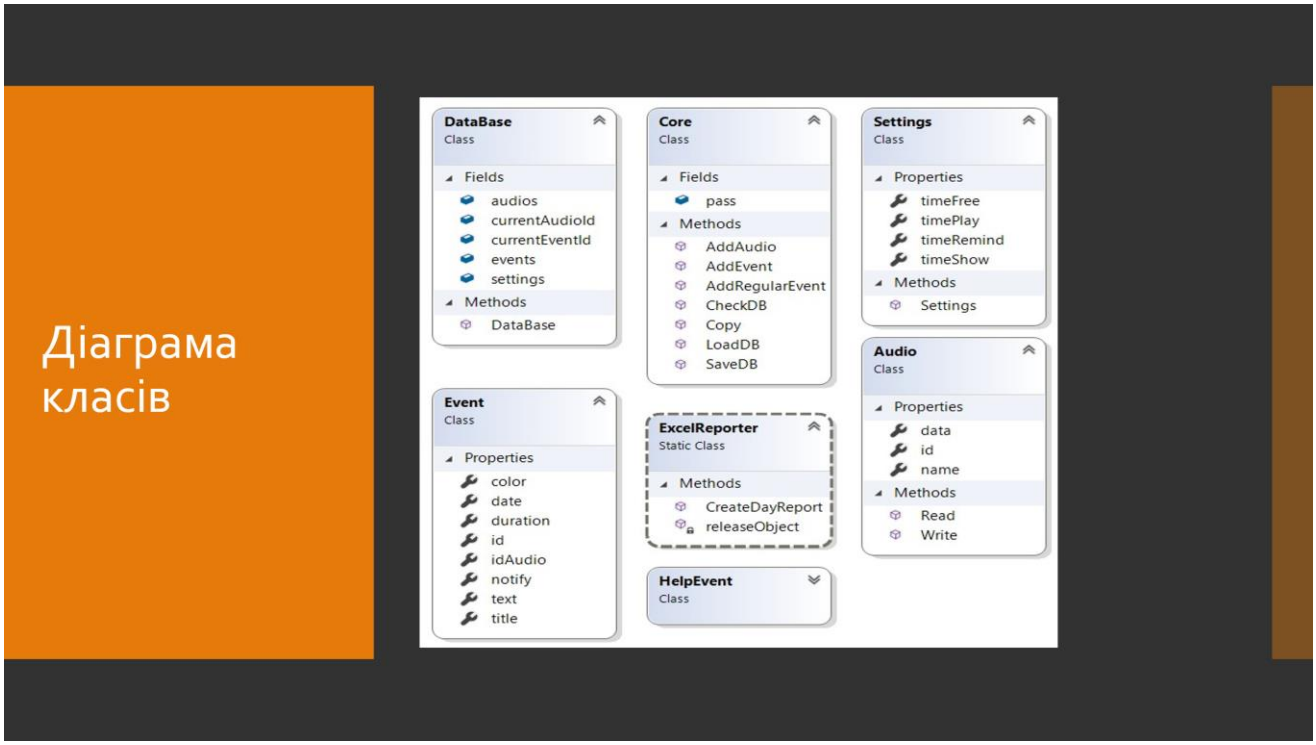

Рисунок Г.2 — Діаграма класів C#

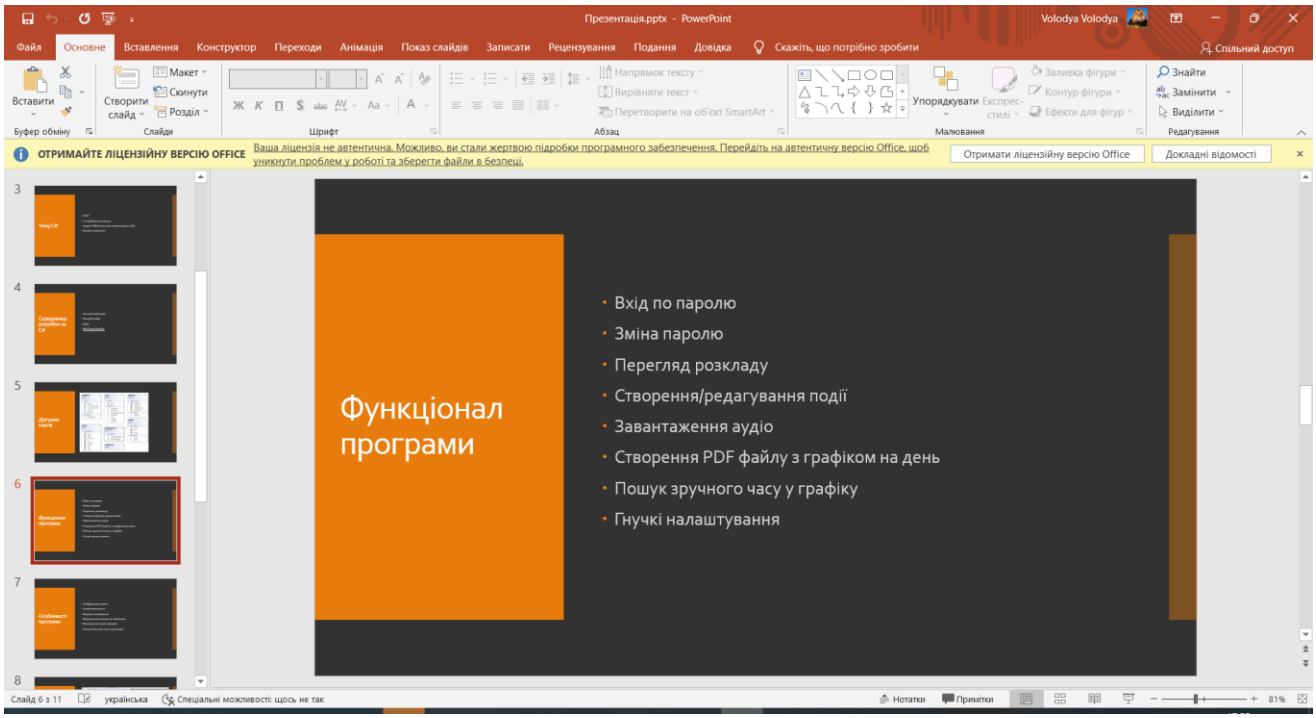

Рисунок Г.3 — Функціонал програми

ДОДАТОК Д

Продовження презентації

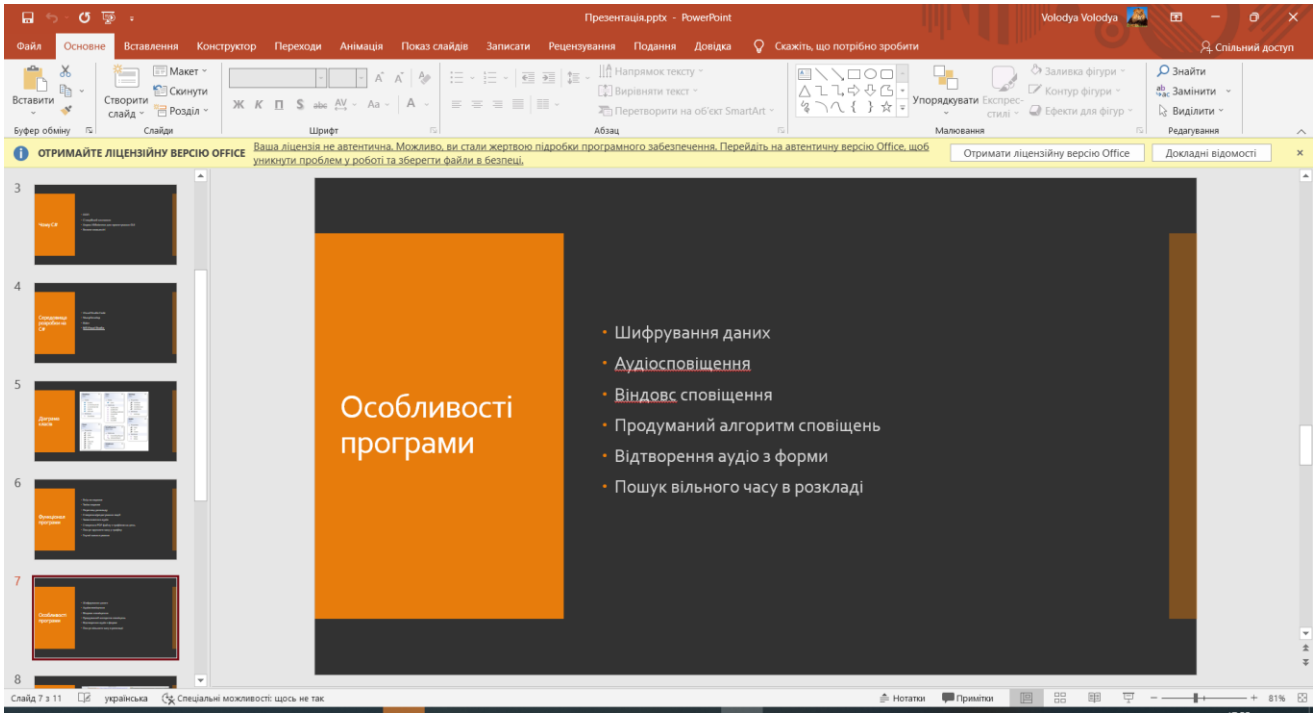

Рисунок Д.1 — Особливості програми

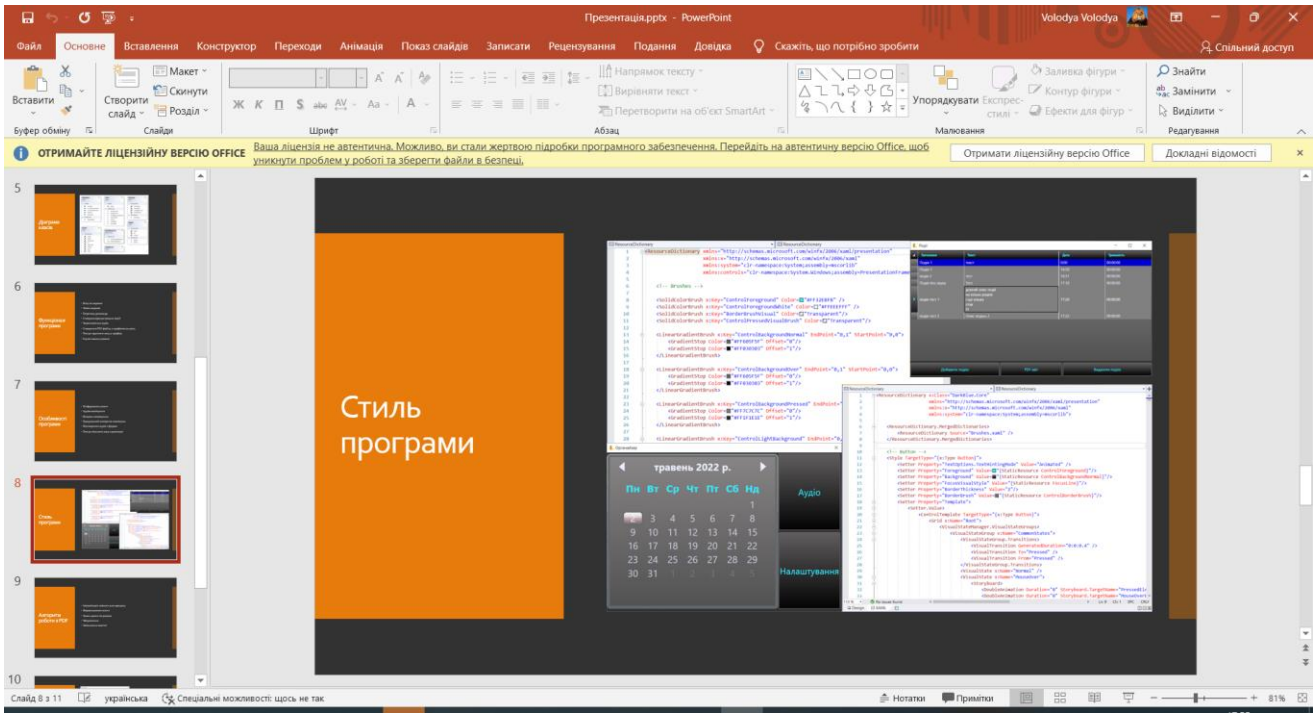

Рисунок Д.2 — Стиль програми

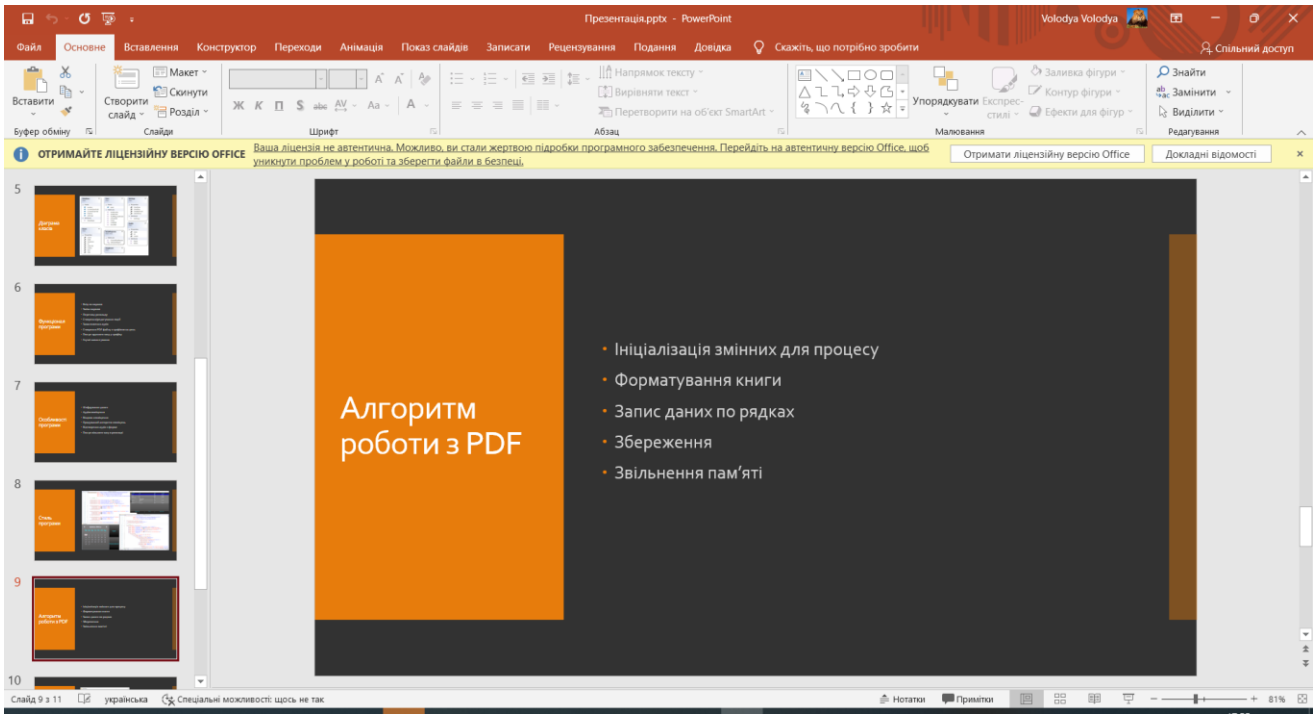

Рисунок Д.3 — Алгоритм роботи з PDF

ДОДАТОК Е

Кінець презентації

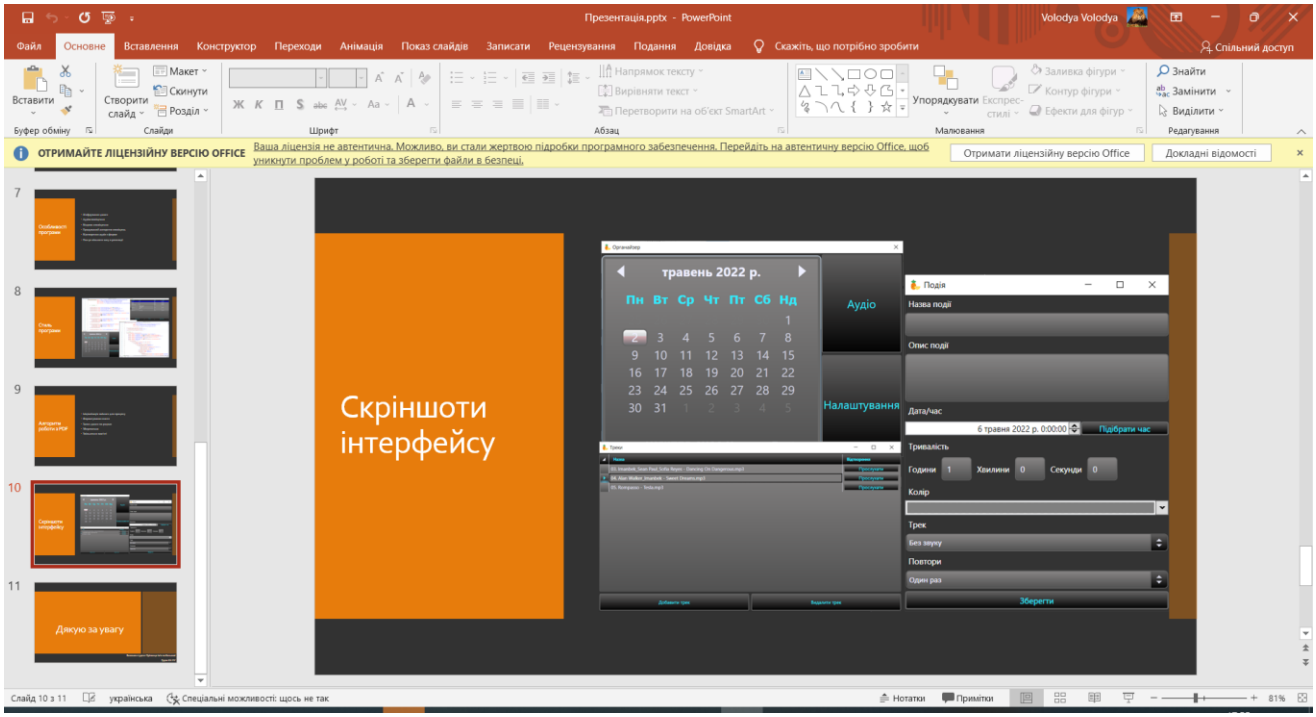

Рисунок Е.1 — Колаж скріншотів інтерфейсу

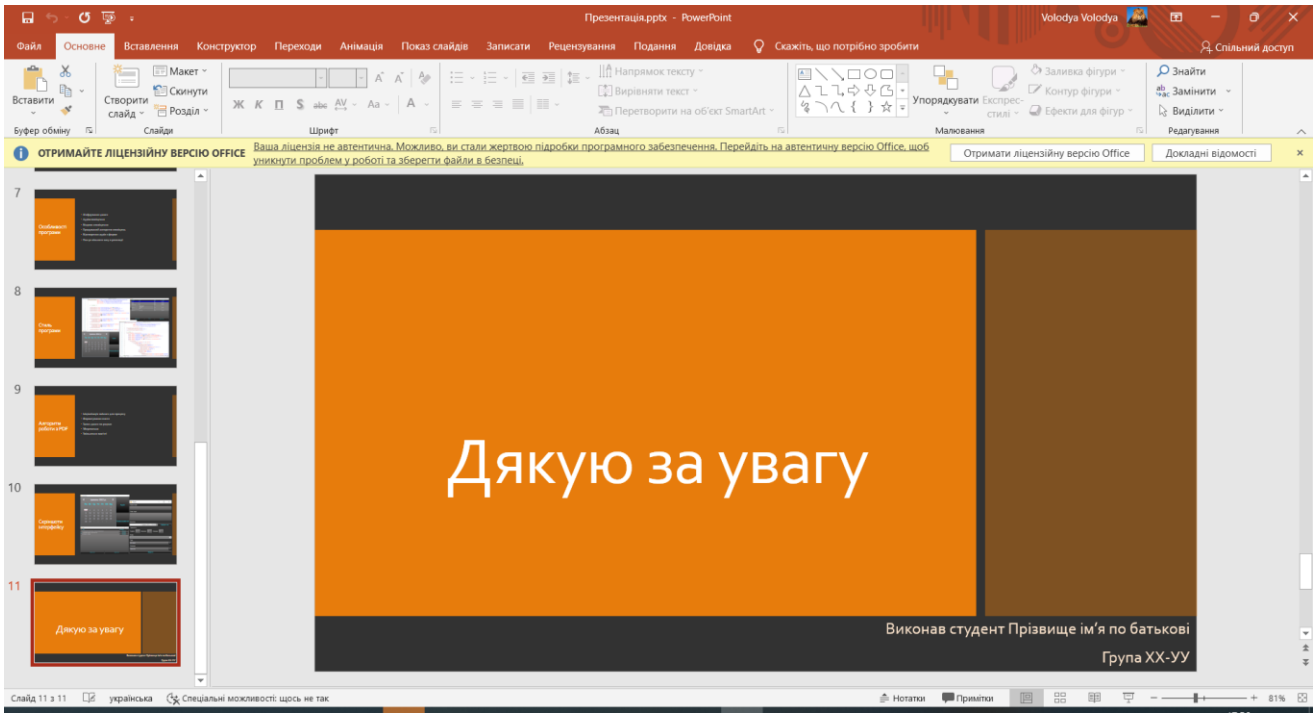

Рисунок Е.2 — Фінальний слайд перезнтації

ДОДАТОК Ж

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: Безпечний органайзер з аудіосповіщенням для навчання

Тип роботи:_____ бакалаврська дипломна робота _____

_____

__

(кваліфікаційна роботи, курсовий проект (робота), реферат, аналітичний огляд, інше (зазначити))

Підрозділ _____кафедра обчислювальної техніки_____

(кафедра, факультет (інститут), навчальна група)

Науковий керівник: доц.каф.ОТ Савицька Л.А.

(прізвище, ініціали, посада)

Показники звіту подібності

Plagiat.pl (StrikePlagiarism)		Unicheck	
КП1			
КП2		Оригінальність	77,4
Тривога/Білі знаки		Схожість	22,6

Аналіз звіту подібності (відмінити подібне)

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності і відсутності самостійності її автора. Робот направити на доопрацювання.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений(-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ ___Вторук О. В._____

(підпис) (прізвище, ініціали)

Опис прийнятого рішення

__ Ступінь оригінальності роботи відповідає вимогам, що висуваються до БР

_____

_____

____

Особа, відповідальна за перевірку_____ _Захарченко С. М.

(підпис) (прізвище, ініціали)

Керівник роботи _____ Савицька Л. А.

(за потреби) (підпис) (прізвище, ініціали)