КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Автоматизації і інформаційних технологій

_____

(факультет)

Інформаційних технологій

_____

(кафедра)

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДО КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ

НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Автоматизована система «Судомодельні змагання»»

Чернишук Олександр Миколайович
(прізвище, ім’я та по батькові студента повністю)

Київ 2024 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Автоматизації і інформаційних технологій

_____
(факультет)

Інформаційних технологій

_____
(кафедра)

ЗАТВЕРДЖУЮ
Завідувач кафедри Інформаційних технологій
Гончаренко Т. А.

_____
„___” _____2024 року

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДО КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ

НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Автоматизована система «Судомодельні змагання»»

Виконав: студент 4–го курсу, групи КНс–21
Спеціальності: 122 «Комп`ютерні науки»
Освітня програма: _Інформаційні управляючі системи і технології.
(шифр і назва напряму підготовки, спеціальності)
_____Чернишук О.М._____.
(прізвище та ініціали)
Керівник _____Горда О.В..
(прізвище та ініціали)
Рецензент _____
(прізвище та ініціали)

Київ 2024 р

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: __автоматизації і інформаційних технологій
Кафедра:__ __інформаційних технологій
Освітній рівень: «бакалавр» за ОП
Спеціальність: _122 «Комп`ютерні науки»
Освітня програма: _ Інформаційні управляючі системи і технології.

ЗАТВЕРДЖУЮ

Завідувач кафедри Інформаційних технологій
Гончаренко Т. А.
_____
„__” ___ __ ___2024 р

ЗАВДАННЯ

ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ

НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

Чернишук Олександр Миколайович

1.Тема роботи: Автоматизована система «Судомодельні змагання»
затверджена наказом ректора КНУБА № 433/2 від «29» лютого 2024 р.
2. Керівник роботи: Горда Олена Володимирівна
кафедри інформаційних технологій проектування і прикладної математики
3. Строк подання студентом роботи до захисту:_____
4. Зміст пояснювальної записки за розділами:
Р.1. Аналіз та дослідження проблеми
Р.2. Проєктуванняінформаційного забезпечення
Р.3. Практична реалізація

5. Інформаційні слайди:
- TDB

6. Календарний план виконання атестаційної випускної роботи

Види робіт та їх зміст	Дата виконання
Р. 1. Аналіз та дослідження проблем	Лютий 2024 р.
Р. 2. Проєктування інформаційного забезпечення забезпечення	Квітень 2024 р.

Р. 3. Практична реалізація	Квітень 2024 р.
Остаточне оформлення роботи	Травень 2024 р.
Направлення роботи на рецензування	Червень 2024 р.
Попередній захист роботи на кафедрі	Червень 2024 р.

8. Дата видачі завдання: 29 лютого 2024 р.

Керівник		Горда О.В.
	(підпис)	(прізвище та ініціали)
Бакалавр		Чернишук О.М.
	(підпис)	(прізвище та ініціали)
Зав. кафедри		Гончаренко Т. А.
	(підпис)	(прізвище та ініціали)

АНОТАЦІЯ

Чернишук О.М. « Автоматизована система "Судомодельні змагання"»

Атестаційна випускна робота бакалавра за спеціальністю: 122 «Комп’ютерні науки», спеціалізація: «Комп’ютерні системи та мережі». – Київський національний університет будівництва і архітектури. – Київ 2024.

Актуальність даної теми полягає в тому, щоб автоматизувати і спростити процес проведення обласних змагань з судомодельного спорту, а також забезпечити об’єктивність оцінювання і надійність проведення. Оскільки наразі змагання проводяться застарілими методами: записи ведуться на папері, а розрахунки здійснюються калькулятором.

SUMMARY

Chernyshuk O.M. "Automated system "Ship model competitions""

Attestation graduation thesis of the bachelor in the specialty: 122 "Computer science", specialization: "Computer systems and networks". - Kyiv National University of Construction and Architecture. – Kyiv 2024.

The relevance of this topic is to automate and simplify the process of holding regional model ship competitions, as well as to ensure the objectivity of the evaluation and the reliability of the conduct. Since the competitions are currently conducted using outdated methods: records are kept on paper, and calculations are made with a calculator.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

СУБД – система управління базами данних

КБ – кілобайт

МБ – мегабайт

ТБ - терабайт

MVC (Model-View-Controller) – модель-представлення – контролер

DFD (Data Flow Diagram) – діаграма потоків даних

ОС – операційна система

ОЗП – оперативно запам’ятовуючий пристрій пристрій

ВСТУП

Всі підприємства зацікавленні в максимізації свого прибутку та мінімізації витрат. Для досягнення таких цілей вони використовують всі можливі методи, одним із яких є використання автоматизованих систем у роботі підприємств.

Програми дозволяють максимально спростити розрахунки, облік документів, автоматизовувати різні процеси та усувати людський фактор. За допомогою додатків можна зберігати та швидко обробляти великі масиви даних. Таким чином, використання програм веде до більш ефективної та беззбійної роботи підприємств.

Для розробки десктопних програм зазвичай використовуються системи візуального програмування, такі як RAD Studio, C++ Builder, Visual Studio та інші. Всі вони мають достатню велику кількість різного інстументарію для швидкого та зручного створення настільних програм.

Для реалізації дипломного проекту було обране середовище розробки Microsoft Visual Studio 2019. Вона має власну бібліотеку розширень, з якої можна легко та зручно встановлюівти різні інструменти для роботи, створені та завантажені до неї іншими програмістами. Крім того, Visual Studio має вмонтовану NuGet — вільну система керування пакунками, що дозволяє зручно додавати до проекту різні бібліотеки.

Для зберігання та обробки даних була обрана Miscosoft Access 2016 — реляційна система управління базами даних (СУБД) корпорації Microsoft. Входить в склад пакету Microsoft Office. Має широкий набір функцій для зручної та ефективної розробки таблиць баз даних та запитів до неї.

Для створення форм був обраний інтерфейс програмування додатків Windows Forms, який є частиною Microsoft .NET Framework. Була використана мова програмування C#.

Мова C# є відносно новою мовою, про яку світу вперше стало відомо тоді, коли Microsoft в липні 2000 р оголосила про вихід першої версії .NET Framework. З тих пір вона сильно виросла в плані популярності і стала чи не найбільш популярною мовою серед розробників Windows додатків, які використовують .NET Framework. Привабливість мови C# напряму пов'язана з її зрозумілим простим та синтаксисом, який походить від синтаксису C/C++, але спрощує деякі речі та дає додаткові функції.

C# - це потужна, об'єктно-орієнтована мова програмування, розроблена компанією Microsoft. Вона є однією з ключових мов для розробки програмного забезпечення на платформі .NET. Ось кілька ключових рис та особливостей мови C#:

Об'єктно-орієнтована: C# підтримує основні концепції ООП, такі як спадкування, поліморфізм, інкапсуляція.

Сильна типізація: Всі змінні в C# мають типи даних, які визначаються під час компіляції. Це дозволяє виявити помилки під час компіляції, що забезпечує більшу надійність програм.

Платформонезалежність: Код, написаний на C#, може бути виконаний на різних платформах, таких як Windows, Linux, а також веб-серверах і мобільних пристроях, за допомогою технологій, таких як .NET Core і Xamarin.

Синтаксис подібний до C/C++/Java: Синтаксис C# схожий на синтаксис інших популярних мов, таких як C, C++ та Java, що полегшує вивчення для розробників, які вже мають досвід роботи з цими мовами.

Можливості мультипарадигмального програмування: Підтримує різні парадигми програмування, такі як процедурне програмування, ООП та функціональне програмування.

Широкий набір бібліотек і фреймворків: C# має велику кількість стандартних бібліотек для різних завдань, а також широкий вибір сторонніх бібліотек і фреймворків

Visual Studio: C# найчастіше розробляється у середовищі розробки Visual Studio, інтегрованій середовищі розробки (IDE) від Microsoft. Visual Studio надає розширені засоби для розробки, налагодження та відлагодження програм на C#.

.NET Framework та .NET Core: .NET Framework - це платформа для розробки та виконання програмного забезпечення, яка підтримує виконання програм на C#. Однак, з виходом .NET Core, з'явилася можливість розробляти та виконувати програми C# на платформах, таких як Linux та macOS.

ASP.NET: Як частина .NET Framework, ASP.NET дозволяє розробляти веб-додатки та веб-сервіси за допомогою мови C#. ASP.NET надає різноманітні інструменти для створення веб-додатків, включаючи підтримку моделі MVC (Model-View-Controller).

Xamarin: Це фреймворк для розробки мобільних додатків, що дозволяє використовувати мову C# для створення додатків для платформ Android та iOS.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 IDE Visual Studio

Visual Studio — це інтегроване середовище розробки (IDE) від компанії Microsoft, яке використовується для створення різноманітних програмних продуктів, включаючи веб-додатки, мобільні додатки, десктопні програми, ігри, вбудовані системи та інші. Ось деякі ключові аспекти та функції Visual Studio:

Мови програмування: Visual Studio підтримує ряд мов програмування, таких як C#, Visual Basic .NET, C++, F# та інші. Це дозволяє розробникам вибирати мову, яка найкраще підходить для їх потреб та навичок.

Інтегровані інструменти розробки: Visual Studio має широкий набір інтегрованих інструментів, які полегшують розробку, відладку, тестування та розгортання програмного забезпечення. Ці інструменти включають редактор коду з підсвічуванням синтаксису, інструменти відлагодження, системи контролю версій, тестові фреймворки та багато іншого.

Підтримка різних платформ: Visual Studio дозволяє розробляти програмне забезпечення для різних платформ, включаючи Windows, macOS, Linux, Android, iOS, Azure, та інші. Це робить його універсальним інструментом для розробки, незалежно від платформи, на яку призначений додаток.

Екосистема розширень: Visual Studio має широку екосистему розширень, які дозволяють розширити його функціональність та налаштувати під конкретні потреби розробника. Це включає розширення для підтримки нових мов програмування, інструментів відлагодження, тем оформлення, інтеграції зі сторонніми сервісами та багато іншого.

Інструменти для колективної роботи: Visual Studio надає інструменти для колективної роботи, такі як системи контролю версій (наприклад, Git), засоби спільної роботи над кодом (наприклад, Live Share), а також

інтеграцію з системами для автоматизації процесів розробки та CI/CD (Continuous Integration/Continuous Deployment).

Інтеграція з облачними сервісами: Visual Studio інтегрується з облачними сервісами Microsoft Azure, що дозволяє розробникам легко створювати, тестувати, розгортати та керувати своїми додатками в хмарному середовищі.

Загалом, Visual Studio є потужним та універсальним інструментом для розробки програмного забезпечення, який надає розробникам широкі можливості та інструменти для ефективної роботи над різноманітними проектами.

Шаблони проектів і кодування: Visual Studio має велику бібліотеку вбудованих шаблонів проектів, які допомагають розпочати роботу з новим проектом швидко та ефективно. Крім того, IDE надає багато корисних функцій, таких як автодоповнення коду, підказки для інтеграції бібліотек, швидкий доступ до документації та інші.

Підтримка тестування: Visual Studio надає розширені засоби для написання та виконання тестів, включаючи єдину платформу для тестування різних типів тестів, таких як модульні, інтеграційні та функціональні тести. Інтегровані інструменти для тестування дозволяють легко створювати, виконувати та аналізувати результати тестів безпосередньо у Visual Studio.

Підтримка розробки мобільних додатків: Visual Studio має інтегровані інструменти для розробки мобільних додатків для платформ Android та iOS. За допомогою розширень, таких як Xamarin, ви можете створювати крос-платформенні мобільні додатки, які працюють на різних мобільних пристроях.

Підтримка навчання і навчання: Visual Studio має багато ресурсів для навчання та навчання, включаючи документацію, уроки, відеоуроки,

онлайн-курси та інші матеріали. Це допомагає розробникам вивчити нові технології, покращити свої навички та знайти відповіді на свої питання.

Visual Studio - це не лише інтегроване середовище розробки, але й екосистема, що надає розробникам широкий набір інструментів, ресурсів та сервісів для ефективної роботи над програмним забезпеченням різного типу та складності.

Налагодження та профілювання: Visual Studio надає потужні інструменти для налагодження програм на C#. Розробники можуть використовувати вбудовані інструменти для аналізу роботи програми, виявлення помилок, відлагодження та профілювання швидкості виконання коду.

Інтегрована підтримка тестування: Visual Studio має різноманітні інструменти для написання, запуску та аналізу тестів у програмах на C#. Це дозволяє розробникам створювати надійне програмне забезпечення за допомогою тестування одиниць, інтеграційного тестування та автоматизованих тестів.

Підтримка колективної роботи: Visual Studio має інтегровані інструменти для роботи у команді. Розробники можуть працювати над спільними проектами, використовуючи системи керування версіями, такі як Git, та інші інструменти для спільної роботи та співпраці.

Навчальні та навчальні ресурси: Microsoft надає багато навчальних та навчальних ресурсів для вивчення мови C# та розробки програм у Visual Studio. Це включає в себе документацію, онлайн-курси, блоги, відеоуроки та інші матеріали, що допомагають розробникам вдосконалювати свої навички.

Підтримка новітніх технологій: Visual Studio постійно оновлюється, щоб підтримувати новітні технології та фреймворки. Це дозволяє розробникам використовувати останні інновації у своїх проектах та створювати сучасні програми.

Широкий вибір шаблонів проектів: Visual Studio має великий набір шаблонів проектів, що дозволяє розробникам швидко створювати нові проекти з необхідною структурою та налаштуваннями. Від настільних програм до веб-додатків і мобільних додатків - ви знайдете відповідний шаблон для вашого проекту.

Підтримка розширеного IntelliSense: IntelliSense - це функціонал Visual Studio, який надає інтелектуальні підказки під час роботи з кодом. Він автоматично виводить доступні методи, властивості та інші частини API під час написання коду, що допомагає прискорити процес розробки та зменшити кількість помилок.

Інструменти для роботи з графікою та дизайном: Visual Studio має інтегровані інструменти для роботи з графічним дизайном, такі як редактор XAML для розробки інтерфейсів користувача для програм на платформі .NET. Це дозволяє розробникам створювати зручні та естетично привабливі користувацькі інтерфейси.

Інтегровані інструменти тестування безпеки: Visual Studio має інтегровані інструменти для тестування безпеки програмного забезпечення. Вони дозволяють розробникам виявляти потенційні проблеми безпеки в коді та виправляти їх до випуску продукту.

Підтримка розширеного відлагодження: Visual Studio має потужні інструменти для відлагодження програмного забезпечення, що дозволяють розробникам аналізувати та виправляти помилки в коді. Це включає в себе можливість крокування по коду, аналізу змін змінних, встановлення точок зупинки та багато іншого.

Visual Studio є повноцінним інструментом для розробки програмного забезпечення на мові C# та інших мовах програмування. Він надає розробникам широкий набір функціональностей та інструментів, що полегшують розробку, тестування та відлагодження програм.

1.2 Платформа .NET FRAMEWORK

.NET Framework — це платформа розробки програмного забезпечення, розроблена компанією Microsoft. Вона надає засоби для створення різноманітних десктопних, веб- та мобільних додатків для операційних систем Windows. Одним з головних призначень .NET Framework є полегшення розробки програм шляхом надання розширеної бібліотеки класів та середовища виконання коду (CLR - Common Language Runtime).

Основні компоненти .NET Framework включають:

CLR (Common Language Runtime): Він відповідає за виконання .NET-коду. CLR відстежує використання ресурсів, керує пам'яттю, обробляє винятки, виконує безпеку та інші завдання, що допомагають забезпечити безпеку та ефективність виконання коду.

Бібліотеки класів .NET (FCL): Це колекція перевірених бібліотек, які надають доступ до різних функцій, таких як робота з файлами, мережею, криптографією, робота з базами даних тощо. Вона значно полегшує розробку, оскільки розробникам не потрібно заново виконувати багато загальних завдань.

Мови програмування: .NET Framework підтримує кілька мов програмування, включаючи C#, Visual Basic .NET, F# та інші. Це дає розробникам можливість вибирати мову, з якою вони найкраще знайомі або яка найбільше відповідає потребам їх проектів.

ASP.NET: Це набір інструментів для розробки веб-додатків, включаючи веб-сторінки, веб-сервіси, а також технології, такі як MVC (Model-View-Controller) та WebForms.

Windows Presentation Foundation (WPF): Це технологія для розробки десктопних додатків з використанням графічного інтерфейсу користувача (GUI). WPF надає багато можливостей для створення сучасних та естетично привабливих додатків.

.NET Framework давно вважається одним з ключових інструментів для розробки програмного забезпечення на платформі Windows, але з появою .NET Core і пізніше .NET 5 і .NET 6, Microsoft поступово переходить до нової платформи .NET, яка пропонує більш широкі можливості, крос-платформенність та відкритий код.

Нова платформа .NET, яка складається з .NET Core, .NET 5 і .NET 6, є важливим кроком у розвитку екосистеми розробки програмного забезпечення від Microsoft. Деякі ключові особливості та переваги нової платформи .NET включають:

Крос-платформенність: .NET Core, базова компонента нової платформи, підтримує розробку та виконання програм на Windows, macOS та Linux. Це дозволяє розробникам створювати додатки, які працюють на різних операційних системах, зменшуючи залежність від конкретної платформи.

Відкритий код: Одним з ключових аспектів нової платформи .NET є перехід до відкритого коду. Відкрита ліцензія сприяє активній спільноті розробників, які можуть спільно працювати над покращенням платформи, вносячи власний внесок у вигляді патчів, допоміжного коду та ідей.

Висока продуктивність: .NET Core та .NET 5/6 надають широкі можливості для оптимізації продуктивності додатків, включаючи використання асинхронного програмування, підтримку мультипроцесорних систем, оптимізовані алгоритми та інші інструменти.

Модульність та гнучкість: Нова платформа .NET дозволяє використовувати тільки ті компоненти, які потрібні для конкретного проекту. Це дозволяє зменшити розмір додатків та полегшує їх розгортання та підтримку.

Широкий вибір інструментів розробки: Нова платформа .NET підтримує використання різних інструментів розробки, включаючи Visual Studio, Visual Studio Code, JetBrains Rider та інші. Це дає розробникам можливість

вибирати інструменти, які найкраще відповідають їх потребам та перевагам.

Загалом, нова платформа .NET є потужним інструментом для розробки програмного забезпечення, яка пропонує широкі можливості, високу продуктивність та гнучкість, а також дозволяє розробникам створювати додатки для різних платформ, зберігаючи при цьому єдність розробки.

Інтеграція з Visual Studio: .NET Framework має глибоку інтеграцію з Visual Studio, основним інтегрованим середовищем розробки (IDE) від Microsoft. Visual Studio надає розробникам широкий набір інструментів для створення, відлагодження та тестування програм на платформі .NET Framework.

Керування версіями: .NET Framework підтримує керування версіями, що дозволяє розробникам використовувати різні версії платформи для різних проектів або навіть для різних частин одного проекту.

Безпека: .NET Framework надає різні механізми безпеки, такі як підписи збірок, код ізоляції, CAS (Code Access Security) та інші, що допомагають забезпечити безпеку програм та захистити систему від шкідливого коду.

Підтримка розробки мультиплатформенних додатків: Хоча .NET Framework спеціалізується на платформі Windows, він також має можливості для розробки мультиплатформенних додатків за допомогою технологій, таких як Mono, який забезпечує підтримку для різних операційних систем, таких як Linux та macOS.

Широке співтовариство та підтримка: Багато розробників, які використовують .NET Framework, діляться знаннями, допомагають один одному та надають різноманітні бібліотеки та інструменти для спільного використання.

Хоча .NET Framework зберігає свою важливість для існуючих проектів і додатків, Microsoft активно розвиває інші версії платформи, такі як .NET

Core і .NET 5 (і пізніше .NET 6 та майбутні версії), які вже стали стандартом для розробки нових додатків.

Платформозалежність: Одним з недоліків .NET Framework є те, що він специфічний для платформи Windows. Хоча Microsoft надає підтримку Mono для Linux та macOS, існують обмеження у функціональності та продуктивності в порівнянні з Windows.

Обмежена підтримка хмарних технологій: Хоча .NET Framework можна використовувати для створення веб-додатків та служб, його підтримка хмарних технологій, таких як мікрослужби або контейнери, не така розгалужена, як у сучасних версіях .NET, таких як .NET Core та .NET 5+.

Підтримка та оновлення: Оскільки .NET Framework є збіркою, яка встановлюється на комп'ютері, вона потребує окремого оновлення від Microsoft. Це може бути не так зручно, як оновлення, що автоматично включені в процес розробки на платформі .NET Core та пізніших версіях.

Спадщина коду та легасі проекти: Оскільки .NET Framework був основною платформою для розробки програм Windows протягом багатьох років, існує велика кількість легасі коду та проектів, які були розроблені для цієї платформи. Переміщення цих проектів на сучасніші версії .NET може бути складним завданням через залежності та обмеження.

У кінцевому підсумку, .NET Framework продовжує залишатися важливою платформою для підтримки існуючих додатків та інфраструктури на платформі Windows. Однак для нових проектів рекомендується розглядати використання сучасних версій .NET, таких як .NET Core або .NET 5+

1.3 Технологія WPF

Windows Presentation Foundation (WPF) - це технологія для створення десктопних застосунків з сучасним графічним інтерфейсом користувача (GUI) в середовищі .NET Framework або .NET Core. Вона була вперше випущена Microsoft у 2006 році як частина .NET Framework 3.0.

Основні особливості технології WPF:

Декларативна розмітка (XAML): WPF використовує мову розмітки XAML (Extensible Application Markup Language), що дозволяє описувати інтерфейс користувача у вигляді розмітки, подій та стилів, подібно до HTML для веб-розробки. Це дозволяє розділити дизайн та логіку програми.

Візуальна стилізація і анімація: WPF надає багаті можливості для стилізації елементів і анімації їх взаємодії. Ви можете використовувати шаблони, стилі, градієнти, трансформації та анімацію, щоб створити сучасний та естетично привабливий інтерфейс користувача.

Розділення дизайну і логіки: WPF підтримує розділення дизайну та логіки програми, що дозволяє розробникам і дизайнерам працювати над проектом паралельно, зменшуючи конфлікти та полегшуючи співпрацю.

Вбудована підтримка графіки та мультимедіа: WPF має вбудовану підтримку векторної та растрової графіки, а також мультимедіа. Це дозволяє створювати додатки з багатошаровими зображеннями, відео та аудіо ефектами.

Можливості масштабування: Використання векторної графіки та розділення дизайну і логіки дозволяє створювати додатки, які легко масштабуються на різних дисплеях та пристроях, забезпечуючи при цьому високу якість зображення.

Підтримка взаємодії з іншими технологіями: WPF добре інтегрується з іншими технологіями Microsoft, такими як Windows Forms, ASP.NET та Windows Communication Foundation (WCF). Це дає можливість створювати повноцінні додатки, які поєднують в собі різні типи інтерфейсів користувача та функціональності.

Підтримка шаблонів і архітектурних патернів: WPF сприяє використанню шаблонів проектування та архітектурних патернів, таких як MVVM (Model-View-ViewModel). Це дозволяє створювати добре

структуровані та легко супроводжувані додатки, розділяючи логіку, відображення та дані.

Можливості налаштування стилів та шаблонів: WPF надає розширені можливості для налаштування стилів та шаблонів елементів інтерфейсу користувача. Ви можете легко змінювати вигляд та поведінку елементів за допомогою CSS-подібних стилів, які визначаються в XAML.

Підтримка даних і зв'язування: WPF надає потужні засоби для роботи з даними та їх зв'язування з елементами інтерфейсу. Ви можете легко зв'язувати дані з об'єктами програми або зовнішніми джерелами даних, такими як бази даних або веб-служби.

Підтримка міжнародних мов і культур: WPF має вбудовану підтримку локалізації та міжнародизації, що дозволяє створювати додатки, які можуть працювати з різними мовами та культурами.

Загалом, WPF є потужною технологією для розробки десктопних додатків у середовищі .NET, що надає розробникам багато можливостей для створення сучасних та функціональних інтерфейсів користувача. Вона поєднує в собі високу продуктивність, гнучкість та широкі можливості налаштування, що робить її популярним вибором для багатьох проектів.

Можливості адаптації до різних пристроїв: WPF надає можливості для створення адаптивних та респонсивних інтерфейсів, які можуть працювати на різних типах пристроїв, включаючи настільні комп'ютери, планшети та мобільні пристрої. Це дозволяє розробникам створювати додатки, які працюють ефективно на різних платформах та розмірах екранів.

Підтримка мультимедіа та 3D-графіки: WPF надає можливості для інтеграції мультимедійних вмісту та 3D-графіки у додатки. Це дозволяє створювати багатофункціональні додатки з високоякісним мультимедійним вмістом та захоплюючими 3D-ефектами.

Підтримка асинхронних операцій: WPF надає можливості для виконання асинхронних операцій у фоновому режимі, що дозволяє уникнути блокування інтерфейсу користувача під час виконання довгих або інтенсивних операцій. Це полегшує створення додатків, які мають високу реактивність та продуктивність.

Розширені можливості доступності: WPF надає розширені можливості для підтримки доступності, що дозволяє створювати додатки, які можуть бути легко використані людьми з обмеженими можливостями. Це включає в себе можливості для встановлення міток, описів та інших елементів, що полегшують використання додатків людьми з різними потребами. Інтеграція з іншими технологіями Microsoft: WPF інтегрується з іншими технологіями Microsoft, такими як .NET Framework, Visual Studio, Expression Blend та інші, що робить його зручним вибором для розробки додатків у середовищі Microsoft.

Технологія WPF є потужним інструментом для створення сучасних та ефективних графічних інтерфейсів користувача для додатків на платформі Windows. Вона надає розробникам широкі можливості для створення динамічних, інтерактивних та привабливих додатків, які здатні вражати та задовольняти потреби користувачів.

Модульність та розширюваність: WPF дозволяє розробникам створювати додатки з модульною структурою, що спрощує управління великими проектами. Компоненти інтерфейсу можуть бути розділені на окремі модулі, які можуть бути розроблені та підтримані незалежно один від одного.

Підтримка адаптивного дизайну і різних форматів екрану: WPF дозволяє розробникам створювати інтерфейси, які можуть адаптуватися до різних розмірів та форматів екранів. Це особливо важливо для додатків, які працюють на різних пристроях з різними розмірами екранів.

Підтримка сторонніх бібліотек і фреймворків: WPF має розширювану архітектуру, що дозволяє розробникам використовувати сторонні бібліотеки та фреймворки для розширення функціональності своїх додатків.

Сумісність з різними версіями Windows: WPF забезпечує сумісність з різними версіями операційної системи Windows, що дозволяє розробникам створювати додатки, які працюють на різних версіях Windows, від Windows 7 до Windows 10.

Підтримка розгортання та установки: WPF має вбудовані засоби для розгортання та установки додатків, що полегшує розповсюдження та встановлення програм на комп'ютерах користувачів.

Технологія WPF залишається важливим інструментом для створення сучасних та ефективних графічних інтерфейсів користувача для Windows-додатків. Вона надає розробникам широкі можливості для створення різноманітних додатків, які здатні вражати та задовольняти потреби користувачів.

1.4 Загальний аналіз предметної області

Згідно із завданням дипломного проекту необхідно розробити програму для зручного та автоматичного проведення обласних судномодельних змагань. Програма повинна передбачати наступний функціонал:

Реєстрація команд і їх учасників;

Обрахунок стендового оцінювання моделей учасників;

Обрахунок ходових випробувань учасників;

Визначення призових місць учасників;

Обрахунок командних балів.

Порядок проведення обласних змагань із судномодельного спорту:

Після прибуття учасники проходять реєстрацію. Спочатку потрібно зареєструвати команду – вказати місто, центр позашкільної освіти, керівника команди. Після чого можна буде зареєструвати учасників. При

реєстрації учасника потрібно вказати наступні дані – прізвище та ім’я учасника, команда, від якої виступає учасник, клас моделі, з якою виступає учасник, залік – моде бути особистий або командний. Коли всі учасники зареєстровані починається стендове оцінювання моделей (якість виконання, зовнішній вигляд), клас F4A не оцінюється. Оцінювання проводять п’ять судей, після виставлення оцінок найменший і найбільший бали ігноруються і обраховується середнє арифметичне із трьох балів для більш точного і чесного цінювання. Після стендового оцінювання проводяться ходові випробування – учасники запускають свої моделі в басейні. Учаснику надається чотири спроби щоб виступити якнайкраще – направити модель у центральні ворота – 100 балів. По мірі проходження ходових випробувань результат спроб записується і по завершенню найгірша спроба ігнорується і обраховується середнє значення із трьох спроб. Після завершення ходових випробувань середні бали учасника за стендове оцінювання та ходові випробування додаються і цей результат враховується як фінальний. За фінальними результатами складається рейтинг і визначаються переможці. Командні місця складаються із суми фінальних результатів учасників які при реєстрації вказали комадний залік. За цією сумою теж складаються рейтингові списки і визначаються командні місця.

Користуватися програмою буде секретар змагань, а всі результати будуть додатково фіксуватися у журналі проведення змагань в паперовому вигляді, що виключає можливість підробки результатів.

1.5 **Опис структурних і функціональних особливостей системи**

У відповідності до теми дипломного, проекту програма повинна включати в себе наступний функціонал:

- Реєстрація команд і учасників;
- Обрахунок і запис стендових оцінок учасників;
- Обрахунок і запис ходових балів за ходові випробування;

- Визначення переможців змагань;
- Обрахунок командних результатів і визначення переможців;
- Виклик довідкової системи про функціонал програми.

У Windows Forms кожна створена форма – це окремий клас, який наслідує клас Form, він містить методи для роботи зі створеною формою. Таким чином, розроблювана програма буде мати наступну структуру:

- Головна форма – форма, на якій будуть розташовані навігаційні кнопки для переходу на інші форми;
- Форма реєстрації учасників – форма, в якій передбачений функціонал для реєстрації нового учасника;
- Форма реєстрації команди – форма, в якій передбачений функціонал для реєстрації команди;
- Форма стендового оцінювання – форма, в якій передбачено функціонал для обрахунків стендових балів (розроблено для всіх класів крім F4A);
- Форма ходових випробувань – форма, в якій передбачено функціонал для обрахунку балів за ходові випробування;
- Форма командного результату – форма, в якій передбачено функціонал для визначення командних місць.

1.6 Доцільність розробки

Розробка десктопної програми для автоматизованого проведення змагань з судомодельного спорту може мати декілька вагомих переваг:

Ефективність організації змагань: Завдяки десктопній програмі можна ефективно організувати різні етапи змагань, такі як реєстрація учасників, формування розкладу змагань, проведення жеребкування, підрахунок результатів тощо. Це дозволяє зменшити час, необхідний для підготовки та проведення змагань, та підвищити їхню організаційну ефективність.

Точність та надійність обробки даних: Використання програмного забезпечення дозволяє уникнути помилок та неточностей при обробці результатів змагань. Система автоматизованого підрахунку результатів може забезпечити надійність та точність результатів, що важливо для справедливого розподілу місць та нагород між учасниками.

Підвищення привабливості змагань для учасників та глядачів: Завдяки використанню сучасних технологій, таких як візуалізація результатів на екрані, інтерактивні таблиці лідерів, онлайн трансляції тощо, можна зробити процес проведення змагань більш захоплюючим і цікавим для учасників та глядачів.

Можливості аналізу та статистики: Десктопна програма може забезпечити можливості для збору та аналізу статистичних даних про учасників та результати змагань. Це дозволяє організаторам отримати більше інформації про характеристики змагань, виявити тенденції та покращити їхню організацію в майбутньому.

Зручність для користувачів: Використання десктопної програми спрощує взаємодію з організаторами змагань для учасників, судей та інших учасників. Вони можуть швидко отримувати доступ до розкладу змагань, своїх результатів, а також зв'язуватися з організаторами через систему сповіщень або чату.

Система реєстрації учасників та управління даними: Десктопна програма може включати зручну систему реєстрації учасників, де вони можуть заповнювати свої дані та реєструватися на змагання. Крім того, програма може забезпечити централізоване управління даними учасників, їхніми результатами та іншою інформацією, що дозволяє зберігати та оновлювати дані ефективно.

Система автоматизованого жеребкування та складання розкладу змагань: Програма може автоматично проводити жеребкування для розподілу учасників на кваліфікаційні групи або змагання та складати розклад змагань

з урахуванням різних факторів, таких як категорії суден, попередні результати тощо.

Інтеграція з системою відеоспостереження: Деякі програми можуть включати можливості інтеграції з системами відеоспостереження, що дозволяє проводити відеофіксацію змагань та забезпечувати більш точне визначення переможців та вирішення спорних ситуацій.

Підтримка різних форматів змагань: Десктопна програма може бути налаштована для підтримки різних форматів змагань, включаючи гонки на швидкість, точність керування, естафетні змагання, демонстраційні покази тощо. Це робить програму універсальним інструментом для організації різних типів змагань.

Можливості аналізу та звітності: Програма може надавати можливості для аналізу даних про змагання, створення звітів та аналітичних даних, що дозволяє організаторам та учасникам отримувати корисну інформацію про хід змагань, тенденції та можливі покращення.

Отже, розробка десктопної програми для автоматизованого проведення змагань з судомодельного спорту може значно покращити організацію, ефективність та привабливість таких змагань для всіх учасників.

1.7 Правила проведення змагань

1.Загальні положення.

1.1.Це положення визначає порядок організації та проведення обласних змагань та конкурсів учнівської молоді з спортивно-технічного моделювання, конструювання та технічних видів спорту напрямку позашкільної освіти в Житомирській області,засновником яких є управління освіти та науки Житомирської обласної державної адміністрації (далі-управління) їх організаційне, методичне і фінансове забезпечення, порядок участі та визначення переможців.

1.2.Обласні змагання та конкурси з спортивно-технічного напрямку проводяться щороку серед учнівської молоді з авіа-, авто-, ракето-,

судномоделювання, операторів колективних радіостанцій, спортивної радіопеленгації, радіоконструкторів та конструкторів повітряних зміїв, картингістів, мотоциклістів та інших напрямків спортивно-технічного моделювання та конструювання (далі-технічної творчості).

1.3.Керівництво підготовкою та проведенням змагань та конкурсів здійснює управління.

1.4.Організаційно-методичне забезпечення та проведення змагань та конкурсів здійснює Комунальний позашкільний навчальний заклад "Житомирський обласний центр науково-технічної творчості учнівської молоді" Житомирської обласної ради (далі-КПНЗ "Облас-
ний центр НТТУМ").

1.5.Для організації та проведення змагань та конкурсів наказами управління затверджуються персональні склади оргкомітетів.

1.6.Склад команд по класах моделей, категоріях техніки, видів конструювання визначається на семінарах з кожного виду змагань та конкурсів, які організовує КПНЗ "Обласний центр НТТУМ" та доводить до відома міських, районних відділів освіти та позашкільних закладів не пізніше ніж за шість місяців до початку проведення цих змагань та конкурсів.

2.Мета, завдання, місце і терміни проведення змагань та конкурсів з технічної творчості.

2.1.Обласні змагання та конкурси проводяться з метою визначення рівня практичної підготовки вихованців гуртків, виявлення, підтримки обдарованої учнівської молоді, залучення її до творчої діяльності та створення умов для самореалізації творчої особистості в сучасному суспільстві.

2.2.Основними завданнями обласних змагань та конкурсів є:

- популяризація технічної творчості серед дітей та учнівської молдоді;
- підвищення рівня технічної та практичної майстерності учасників;

- пропаганда та популяризація науково-технічних знань серед школярів;

- задоволення потреб дітей і підлітків у професійному самовизначенні відповідно до їхніх інтересів та здібностей;

- розширення мережі гуртків спортивно-технічного напрямку в навчальних закладах.

2.3.Дата, місце та час проведення обласних конкурсів та змагань визначаються наказами управління.

3.Умови участі в конкурсах та змаганнях.

3.1.В обласних змаганнях та конкурсах беруть участь команди, що складаються з вихованців позашкільних навчальних закладів, а також учнів загальноосвітніх шкіл, які займаються технічною творчістю самостійно та за станом здоров'я можуть бути допущені до участі в змаганнях та конкурсах.

3.2.Кількісний склад команди від району, міста, позашкільного навчального закладу не обмежується. При умові, що команда представлена в повному і більше складі, кожен член команди може виступати в двох класах моделей. Не допускається представлення колективних робіт,а також не допускається заміна представництва за класами моделей після подання заявки в оргкомітет.

3.3.Змагання та конкурси проводяться згідно відповідних діючих Правил.

3.4.КПНЗ"Обласний центр НТТУМ" розробляє програму для кожного виду змагань та конкурсів і не пізніше ніж за місяць до їх початку доводить до відома міських та районних відділів освіти та позашкільних закладів.

3.5.Не пізніше ніж за 14 днів до початку змагань та конкурсів повинна бути подана попередня заявка на участь в даному заході на електронну адресу КПНЗ"Обласний центр НТТУМ" .

3.6.До місця проведення обласних змагань та конкурсів команди прибувають у супроводі керівника-педагогічного працівника, який брав активну участь у підготовці учнів до змагань. Керівник команди повинен

знати і виконувати вимоги даного Положення, офіційних Правил змагань та програми проведення змагань та конкурсів.

3.7.Керівник команди забезпечує:

- ознайомлення членів команди з даним Положенням, діючими Правилами Програмою змагань(конкурсів), дотримання їх вимог;

- своєчасне оформлення необхідних документів;

- прибуття учасників до місця змагань та конкурсів і повернення їх до місця проживання;

- необхідну морально-психологічну підтримку та допомогу учасникам;

- безпеку життя та здоров'я учасників у дорозі та під час проведення змагань та конкурсів;

- захист прав та інтересів учасників крманди;

- дотримання членами команди норм поведінки, дисципліни, правил техніки безпеки та охорони здоров'я, чесної спортивної боротьби.

3.8.Керівник команди при реєстрації подає в оргкомітет такі документи:

- копію наказу відділу(управління) освіти про підсумки проведення змагання або конкурсу в районі(місті) або аналітичний звіт;

- заявку на участь узмаганні або кониурсі(додаток №2);

- паспорти (для учасників віком до 16 років – свідоцтво про народження) на кожного учасника(з поверненням);

- учнівські квитки або довідки з фотографією, завірені печаткою і підписом директора закладу освіти(з поверненням).

3.9.На моделі, технічні засоби, прилади та ін. керівник надає документацію згідно вимог Правил конретного виду технічної творчості.

4.Порядок підведення підсумків.

4.1.Під час обласних змагань та конкурсів визначається командна та особиста першість.

4.2.Особиста першість визначається в кожному класі моделей чи виду технічної творчості при участі в ньому не менше 3 учасників за

максимальною кількістю балів, набраних учасником. Якщо в даному класі моделей 3 учасники, то нагороджується лише 1 місце, якщо 4 учасники – то 1 та 2 місця, якщо 5 і більше учасників – то 1, 2 та 3 місця.

При однаковій кількості балів, набраних кількома учасниками, для визначення місця перевага надається учаснику, який набрав більшу кількість балів на ходових змаганнях, або проводяться перегони згідно діючих Правил.

4.3.Командна першість визначається за максимальною сумою балів, набраних учасниками команди у всіх класах моделей (виду творчості) під час змагань чи конкурсу. Якщо кількість команд складає 3, то нагороджується лише 1 місце, якщо 4 команди – то 1 та 2 місця, якщо 5 та більше – то 1, 2 та 3 місця.

При однаковій кількості балів, набраних командами, перевага віддається тій команді, яка має більшу кількість призових місць в особистих заліках.

4.4.Якщо учасник виступає в двох класах моделей, до командного заліку береться один із двох результатів.

5.Нагородження переможців.

5.1Учасники змагань та конкурсів, які зайняли 1, 2 та 3 місця, нгороджуються дипломами відповідних ступенів.

5.2.Команди,які зайняли 1. 2 та 3 місця, нагороджуються дипломами відповідних ступенів.

5.3.Результати обласних змагань та конкурсів затверджуються наказами.

6.Фінансове забезпечення.

6.1.Витрати на організацію та проведення обласних змагань та конкурсів відносяться за рахунок КПНЗ “Обласний центр НТТУМ” відповідно до затвердженого кошторису.

6.2.Оплата праці оргкомітетів (журі, суддівських бригад) проводиться відповідно до чинного законодавства на договірних підставах.

6.3.Витрати на проїзд учасників та керівників команд, їх харчування під час змагань та конкурсів, ночівлю, відрядження суддів та членів журі здійснюють організації, які їх відряджають.

1.8 Оцінювання моделей

Стендове оцінювання судомоделей - це метод оцінювання, який використовується для оцінки якості виготовлення та деталізації моделей суден. Під час такого оцінювання учасники змагань презентують свої моделі перед журі або суддями, які оцінюють їх за різними критеріями, такими як:

Деталізація і реалістичний вигляд: Цей критерій оцінює, наскільки добре модель відтворює реальний аналог. Він включає в себе якість виконання деталей судна, подачу, реалістичність фарбування та обробки, наявність додаткових елементів, таких як малюнки, написи, емблеми тощо.

Точність масштабування: Цей критерій оцінює, наскільки вірно модель відтворює реальний судновий аналог за масштабом. Він включає в себе відповідність розміру, пропорційність та деталізацію моделі в порівнянні з оригіналом.

Якість виконання та конструкції: Цей критерій оцінює, наскільки добре зроблена та збалансована конструкція моделі. Він включає в себе якість використаних матеріалів, дотримання технічних стандартів, стійкість та міцність конструкції.

Оригінальність та творчий підхід: Цей критерій оцінює, наскільки оригінальною та творчою є модель, наскільки вона відрізняється від інших та має унікальні особливості або додаткові деталі, які роблять її особливою.

Стендове оцінювання дозволяє суддям та журі змагань оцінити якість та творчий підхід учасників, а також визначити переможців у категоріях, пов'язаних з деталізацією та завнішнім виглядом судомоделей. Воно стимулює учасників до удосконалення своїх навичок та креативності у

виготовленні моделей та додає елемент справедливої конкуренції до судомодельного спорту.

Масштабна швидкість моделей - це поняття, що використовується для визначення реальної швидкості об'єкта відносно його реального прототипу за масштабом. У судомодельному спорті це поняття дуже важливе, оскільки моделі суден часто масштабуються відносно реальних суден, і швидкість їх руху також масштабується.

Наприклад, якщо модель судна масштабується відносно реального судна у співвідношенні 1:10, це означає, що кожен сантиметр довжини моделі відповідає 10 сантиметрам реальної довжини судна. Так само, якщо реальне судно рухається зі швидкістю 20 км/год, то масштабна швидкість моделі буде дорівнювати 200 км/год (20 км/год * 10).

У судомодельному спорті швидкість моделей може бути виміряна за допомогою спеціальних приладів, таких як радари, які дозволяють точно виміряти швидкість руху моделі на воді. Ці дані використовуються для оцінки продуктивності та ефективності моделі, а також для порівняння швидкості моделі з швидкістю реального судна.

Масштабна швидкість моделей є важливим аспектом судомодельного спорту, оскільки вона дозволяє учасникам змагань оцінити ефективність та швидкість їхніх моделей порівняно з реальними суднами, а також дозволяє визначити переможців у змаганнях на основі їхніх результатів.

2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Постановка задачі

Відповідно до завдання дипломного пректу необхідно розробити автоматизовану систему визначення рейтингів учасників гуртка технічної творчості.

Необхідно дослідити задану предменту область дипломного проекту та на основі аналізу визначити структуру та архітектуру програмного застосунку, середовище розробки, мову програмування та струтуру, модель даних, інтерфейс програмних модулів та їх взаємозв’язок, довідкову систему та засоби захисту. Також необхідно визначити технічні параметри та мінімальні системні вимоги, необхідні для беззбійної роботи додатку.

Програма повинна виконувати такі функції:

- реєстрацію команд і учасників;
- обрахунок і запис стендового оцінювання;
- обрахунок і запис ходових випробувань;
- визначення переможців (учасників і команд);
- Виклик довідкової системи з використання програми;

2.2 Інформаційне забезпечення проектної системи

Інформаційне забезпечення проектної системи - це сукупність методів, інструментів і процесів, які використовуються для збору, аналізу, збереження, обробки та поширення інформації в рамках проекту. Його мета полягає в тому, щоб забезпечити ефективну комунікацію та управління інформацією всередині команди проекту, замовників та інших зацікавлених сторін.

Інформаційне забезпечення проектної системи включає в себе:

Системи управління проектами (Project Management Systems): Це програмні засоби для планування, організації, виконання та контролю

проектних завдань. Вони забезпечують можливість створення графіків, списків завдань, ресурсного планування тощо.

Електронні засоби комунікації (Electronic Communication Tools): Це інструменти для обміну інформацією між учасниками проекту, такі як електронна пошта, чати, форуми, спільні документи тощо.

Системи збереження та обміну даними (Data Storage and Sharing Systems): Вони дозволяють зберігати і доступатися до документів, файлів, специфікацій та інших матеріалів проекту в централізованому або розподіленому середовищі.

Системи звітності та аналізу (Reporting and Analysis Systems): Ці інструменти допомагають створювати звіти про прогрес проекту, а також аналізувати дані для виявлення тенденцій та проблем.

Засоби відстеження версій (Version Control Tools): Вони дозволяють керувати версіями програмного забезпечення, документів та інших файлів, забезпечуючи можливість відслідковувати зміни та відновлювати попередні версії.

Засоби безпеки і захисту інформації (Security and Information Protection Tools): Це системи для захисту конфіденційної інформації проекту від несанкціонованого доступу, такі як шифрування даних, системи аутентифікації та авторизації, антивірусне програмне забезпечення тощо.

Загальна мета інформаційного забезпечення проектної системи полягає в забезпеченні доступності, цілісності, конфіденційності та достовірності інформації, а також у підтримці ефективності та успішності виконання проекту.

Системи управління ресурсами (Resource Management Systems): Ці системи допомагають у плануванні та використанні ресурсів проекту, таких як людські ресурси, обладнання, матеріали тощо. Вони допомагають у визначенні потреб у ресурсах, розподілі ресурсів між завданнями та моніторингу використання ресурсів.

Системи управління ризиками (Risk Management Systems): Ці системи допомагають ідентифікувати, аналізувати та керувати ризиками проекту. Вони дозволяють оцінювати ймовірність виникнення ризиків, визначати їх вплив на проект та розробляти стратегії мінімізації ризиків.

Системи контролю якості (Quality Control Systems): Ці системи допомагають забезпечити відповідність результатів проекту встановленим стандартам та вимогам. Вони включають інструменти для контролю якості виробів або послуг, виявлення невідповідностей та впровадження виправних заходів.

Системи управління змінами (Change Management Systems): Ці системи допомагають ефективно керувати змінами в проекті, включаючи зміни в обсязі робіт, графіку, витрати, ресурси тощо. Вони допомагають виявляти потреби в змінах, оцінювати їх вплив та контролювати їх виконання.

Системи управління відносинами з клієнтами (Customer Relationship Management Systems): Ці системи допомагають у взаємодії з клієнтами або стейкхолдерами проекту. Вони дозволяють відстежувати комунікацію з клієнтами, враховувати їхні вимоги та очікування та забезпечувати їх задоволеність результатами проекту.

Системи управління витратами (Cost Management Systems): Ці системи допомагають у плануванні, контролі та моніторингу витрат проекту. Вони дозволяють встановлювати бюджет, відстежувати витрати, виявляти перевитрати та розробляти стратегії зниження витрат.

Системи управління інтеграцією (Integration Management Systems): Ці системи допомагають координувати всі аспекти проекту та забезпечувати їхню взаємодію. Вони дозволяють забезпечити консистентність та цілісність проекту.

Системи управління реалізацією (Implementation Management Systems): Ці системи допомагають у плануванні та контролі за фактичним

виконанням завдань і процесів проекту. Вони дозволяють відстежувати прогрес виконання, виявляти затримки та розробляти плани виправлення.

Системи управління термінами (Time Management Systems): Ці системи допомагають у плануванні та контролі за графіком виконання проекту. Вони дозволяють встановлювати терміни завдань, відстежувати прогрес та вчасно реагувати на затримки.

Кожна з цих систем відіграє важливу роль у керуванні проектом і забезпечує необхідну інформацію та інструменти для досягнення успіху. Інформаційне забезпечення проектної системи є основою для ефективного управління проектом та досягнення його цілей.

2.2.1 Структура і схеми інформаційних об’єктів і ресурсів

Об'єкти інформації:

Документи: Включають будь-які письмові, електронні або інші види документів, що містять інформацію про проект.

Дані: Конкретні факти, цифри, вимірювання або структурована інформація, яка використовується для прийняття рішень.

Медіа-файли: Фотографії, відео, аудіо або інші медіа, які можуть бути пов'язані з проектом.

Ресурси інформаційного забезпечення:

Системи управління проектом (Project Management Systems): Програмні засоби для планування, контролю та моніторингу проекту.

Бази даних: Сховище структурованої інформації, яке може бути використане для зберігання та організації даних проекту.

Інструменти звітності: Програмні засоби для генерації звітів та аналізу даних проекту.

Схеми інформаційного потоку:

Схема документообігу: Описує, як документи пересилаються та обробляються в рамках проекту.

Схема звітності: Визначає, як і коли генеруються та розповсюджуються звіти про проект.

Схема комунікації: Описує, як відбувається обмін інформацією між учасниками проекту.

Метадані:

Метадані документів: Інформація про документи, така як автор, дата створення, ключові слова тощо.

Метадані даних: Інформація про дані, яка допомагає їх ідентифікувати та описувати.

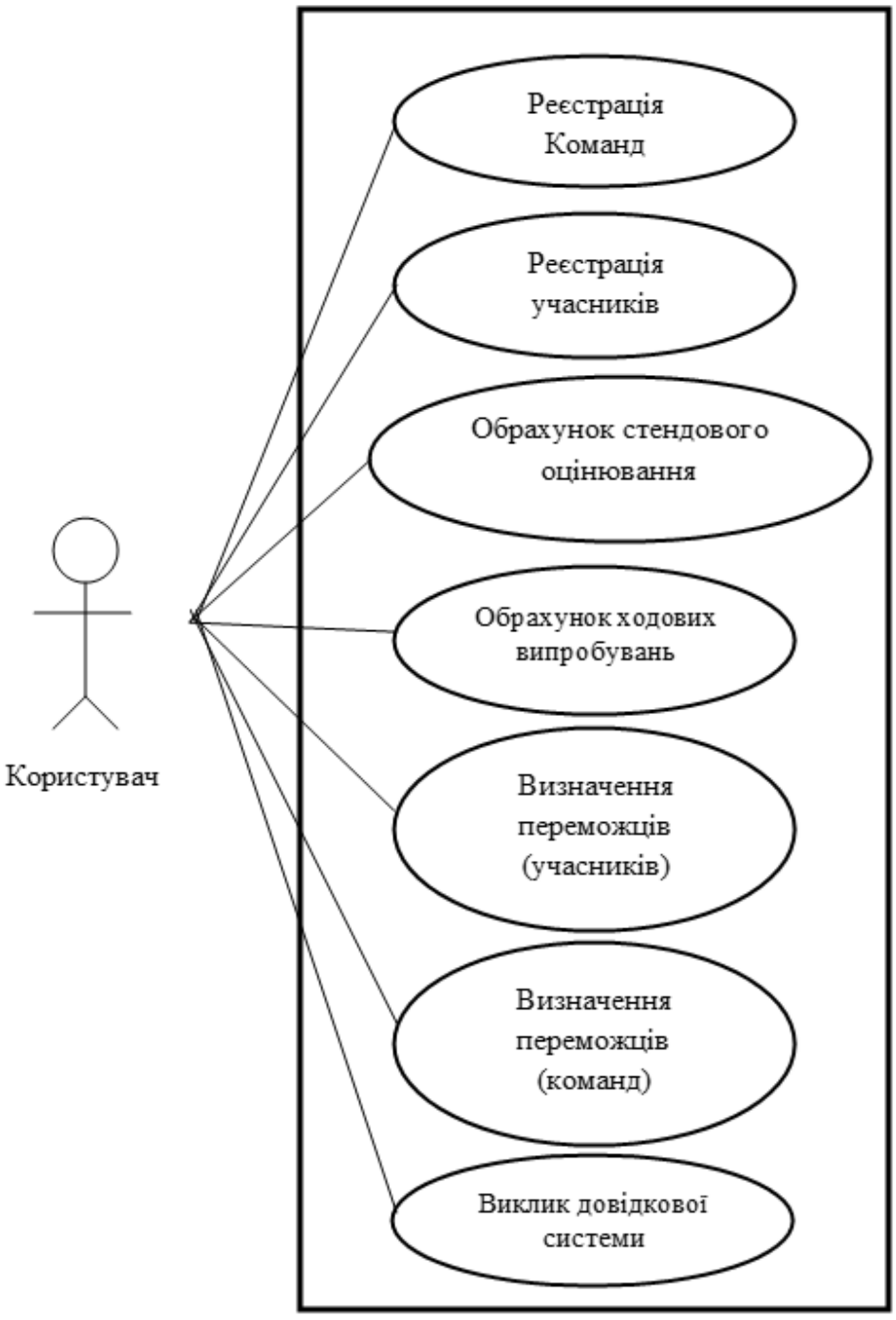
Ці елементи структури і схем інформаційних об'єктів і ресурсів допомагають організувати та управляти інформацією в рамках проекту або системи, забезпечуючи доступність, цілісність та ефективне використання даних.

Розробку специфікацій програмного забезпечення починають з аналізу вимог до функціональності, вказаних в технічному завданні. В процесі аналізу виявляють зовнішніх користувачів програмного забезпечення, що розробляється, і перелік окремих аспектів його поведінки в процесі взаємодії з конкретними користувачами. Аспекти поведінки програмного забезпечення називають «варіантами використання» або «прецедентами» (use cases).

Кожен варіант використання пов'язаний з деякою метою, що має самостійне значення. Діаграми варіантів використання дозволяють наочно представити очікувану поведінку системи. Основними поняттями діаграм варіантів використання є: дійова особа, варіант використання, зв'язок.

Дійовими особами системи є користувач. Варіанти використання виявляємо, аналізуючи технічне завдання, і зображуємо на діаграмі, пов'язуючи з відповідними діючими особами.

Рисунок 2.1 Діаграма варіантів використання – UML – візуальне представлення графу використання представлена на рисунку 2.1.



Діаграма варіантів використання (Use Case Diagram) є одним із типів діаграм в мові моделювання UML (Unified Modeling Language). Вона використовується для візуалізації функціональності системи з точки зору того, як користувачі будуть взаємодіяти з системою або які задачі вони можуть виконати.

У діаграмі варіантів використання головною одиницею є "варіант використання" або "сценарій", який представляє собою конкретний спосіб взаємодії користувача з системою для досягнення певної мети. Кожен варіант використання описується умовною горизонтальною еліптичною фігурою, яка містить назву варіанту використання.

У варіантів використання можуть бути використані різні ролі (actors), які представляють зовнішніх сутностей, такі як користувачі, системи або зовнішні сервіси, які взаємодіють з системою. Ролі зображуються зовнішніми сутностями за допомогою прямокутників.

Взаємозв'язки між ролями та варіантами використання показують способи, якими ролі взаємодіють з системою для виконання конкретних дій.

Ця діаграма допомагає команді розробників та зацікавленим сторонам зрозуміти функціональність системи та забезпечити відповідність її потребам користувачів.

Крім того, діаграма варіантів використання UML може бути використана для:

Уточнення вимог: Діаграма дозволяє ідентифікувати всі можливі способи взаємодії користувачів з системою, що допомагає уточнити та розширити вимоги до системи.

Планування функціоналу: На основі варіантів використання можна розробити план функціоналу системи, визначивши, які функції повинні бути реалізовані для кожного варіанту використання.

Комунікація зі зацікавленими сторонами: Діаграма надає зрозуміле та лаконічне зображення того, як система буде використовуватися користувачами, що сприяє ефективній комунікації з різними зацікавленими сторонами.

Виявлення можливих ризиків та недоліків: Аналіз варіантів використання допомагає ідентифікувати можливі проблеми та ризики, пов'язані з

функціоналом системи, що дозволяє їх вирішити ще на ранніх етапах розробки.

Планування тестування: На основі варіантів використання можна розробити план тестування системи, визначивши, які сценарії потрібно випробувати для перевірки коректності роботи системи.

Отже, діаграма варіантів використання є важливим інструментом для аналізу, проектування та розробки програмного забезпечення, який допомагає забезпечити успішну реалізацію функціоналу системи та задоволення потреб її користувачів.

2.2.2 Схеми інформаційних потоків

Схеми інформаційних потоків - це візуальні зображення, які відображають шляхи руху інформації в системі або організації. Ці схеми допомагають зрозуміти, як дані передаються, обробляються та зберігаються всередині системи. Вони можуть включати в себе різноманітні компоненти, такі як вхідні та вихідні точки, процеси обробки, зберігання даних, а також шляхи комунікації між ними.

Такі схеми можуть бути корисними при аналізі та вдосконаленні процесів управління інформацією в організаціях, розробці нових програмних продуктів або впровадженні змін в існуючі системи. Вони допомагають виявити потенційні проблеми з ефективністю або безпекою, а також забезпечують зрозуміння загальної архітектури системи. Схеми інформаційних потоків можуть бути створені у різних форматах, таких як блок-схеми, діаграми потоку даних (DFD), схеми синхронізації, діаграми Ганта та інші. Кожен з цих форматів може надати певні переваги залежно від конкретної ситуації та потреб аналізу.

Блок-схеми використовуються для візуального представлення послідовності операцій або процесів. Діаграми потоку даних (DFD) спрямовані на зображення потоків даних в системі та взаємодії між різними компонентами. Схеми синхронізації можуть бути корисними для відображення паралельних процесів та умов взаємодії між ними. Діаграми Ганта допомагають візуально відслідковувати часові рамки та послідовність завдань у проекті.

Створення таких схем допомагає уточнити процеси, з'ясувати проблеми та забезпечити краще розуміння того, як працює система. Вони є важливим інструментом для аналізу, проектування та управління інформаційними потоками в будь-якій сфері діяльності.

Інформаційна модель — модель автоматизованої системи, представлена у вигляді інформації, що зображує істотні для даного розгляду параметри та змінні величини об'єкта, зв'язки між ними, входи і виходи об'єкта і дозволяє шляхом подачі на модель вхідних величин моделювати можливі стани об'єкта. Може бути представлена у вигляді функціональної схеми, наведеної на рисунку 2.2.

Діаграма потоків даних (англ. Data Flow Diagram) — так називається методологія графічного структурного аналізу, що описує зовнішні по відношенню до системи джерела і адресати даних, логічні функції, потоки даних і сховища даних, до яких здійснюється доступ. Діаграма потоків даних для функції перегляду формуляра читача (за нотацією Йордана) зображена на рисунку 2.3.

Рисунок 2.2 – Інформаційна модель системи

Інформаційна модель системи - це абстрактне уявлення про те, як дані та інформація обмінюються, обробляються та зберігаються всередині системи. Це концептуальне представлення структури та взаємозв'язків між різними компонентами системи, яке дозволяє зрозуміти, як система функціонує.

Інформаційна модель може включати в себе такі елементи:

Сутності (Entities): Це основні об'єкти, які представляють реальні або абстрактні об'єкти, про які збирається, обробляється або зберігається інформація. Наприклад, користувачі, продукти, замовлення тощо.

Атрибути (Attributes): Це властивості сутностей, які описують характеристики цих об'єктів. Наприклад, для сутності "користувач" атрибутами можуть бути ім'я, прізвище, адреса тощо.

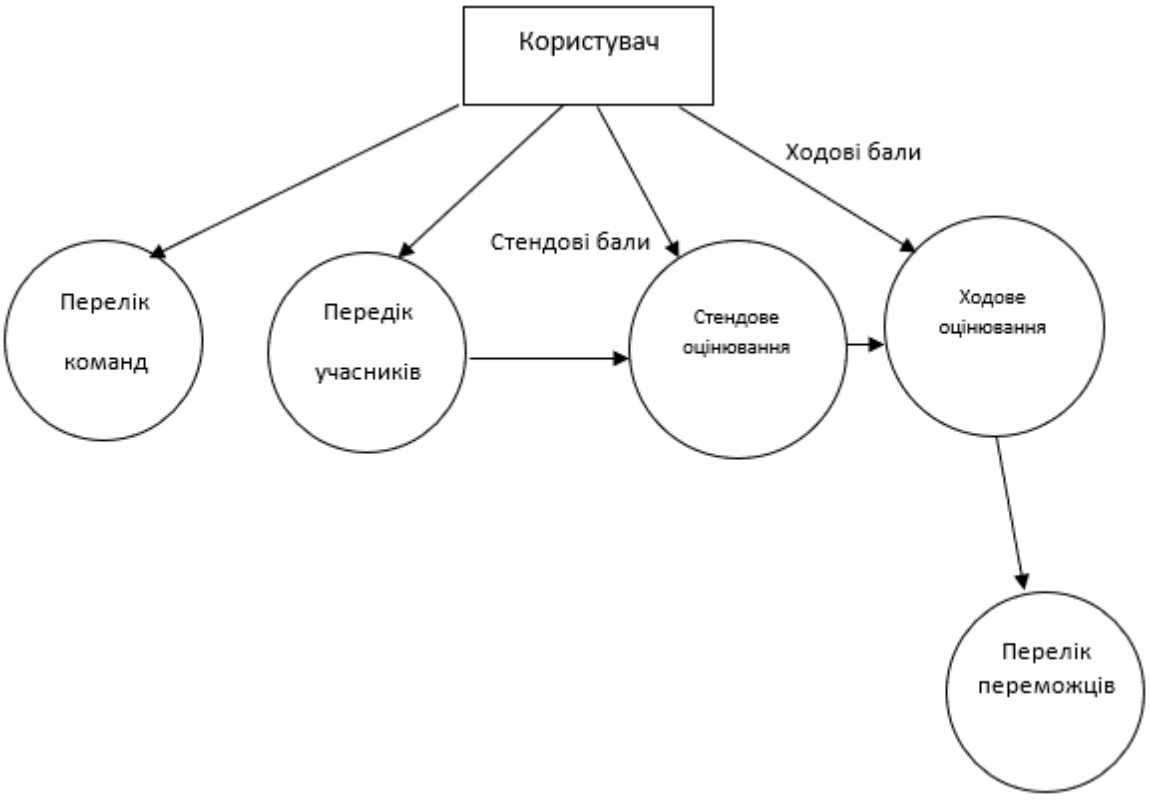
Взаємозв'язки (Relationships): Це зв'язки між різними сутностями, які показують, як вони пов'язані між собою. Наприклад, замовлення пов'язані з користувачами, які їх розміщують.

Процеси (Processes): Це дії або операції, які відбуваються всередині системи для обробки або зміни інформації.

Потоки даних (Data Flows): Це шляхи, по яких дані та інформація переміщаються всередині системи між сутностями, процесами та зберіганням даних.

Створення інформаційної моделі допомагає розібрати складність системи та забезпечити чітке розуміння її структури та функціоналу. Це важливий етап в процесі розробки системи, оскільки він дозволяє забезпечити взаєморозуміння між різними учасниками проекту та забезпечити відповідність вимогам та очікуванням замовника.

Рисунок 2.3 – Діаграма потоків даних (DFD діаграма)



Діаграма потоків даних (DFD) - це графічне зображення потоків даних через систему. Вона складається з різних рівнів деталізації, які демонструють, як дані переходять від початкових вхідних точок, обробляються в системі та виходять у вигляді результату.

DFD може містити наступні елементи:

Процеси (Processes): Це операції або функції, які виконуються над даними для їх обробки або трансформації.

Потоки даних (Data Flows): Це шляхи, по яких дані переміщаються від одного місця до іншого всередині системи.

Сховища даних (Data Stores): Це місця, де дані зберігаються на певний час, зазвичай бази даних або файлові системи.

Вхідні та вихідні точки (External Entities): Це зовнішні об'єкти або системи, які взаємодіють з системою, вводячи або виводячи дані.

DFD діаграма допомагає візуально зрозуміти, як працює система, як дані переміщуються через неї та як вони обробляються. Вона дозволяє ідентифікувати основні компоненти системи та їх взаємозв'язки. Це може бути корисним для аналізу, проектування та вдосконалення інформаційних систем.

Звісно, ось ще деякі ключові моменти, які варто врахувати при створенні DFD:

Рівні деталізації (Level of Detail): DFD може бути створена на кількох рівнях деталізації. Починаючи з високорівневого огляду всієї системи та поступово деталізуючи окремі елементи, щоб отримати більш детальний опис процесів та потоків даних.

Контроль доступу (Access Control): DFD може відобразити механізми контролю доступу до даних та процесів у системі, показуючи, хто має доступ до якої інформації.

Додаткові елементи: У складних системах DFD може включати інші елементи, такі як умовні обробники, які показують, що відбувається в певних умовах, або анотації для пояснення окремих аспектів діаграми.

Ідентифікація інтерфейсів: DFD також може використовуватися для визначення взаємодій з іншими системами або сервісами через зовнішні інтерфейси.

В цілому, DFD є потужним інструментом для аналізу та проектування інформаційних систем. Вона дозволяє розуміти, як дані пересуваються через систему, як вони обробляються та зберігаються, і є основою для розробки більш детальних моделей системи.

2.2.3 Опис та модель даних

База даних - це організована колекція даних, яка зберігається та управляється з метою ефективного доступу, оновлення та аналізу. Вона служить для зберігання інформації, яка може бути доступна для різних користувачів або програм. Бази даних використовуються в різноманітних сферах, включаючи бізнес, науку, організаційне управління та багато іншого.

Основні компоненти бази даних включають:

Дані: Це фактична інформація, яка зберігається в базі даних. Вони можуть бути в різних форматах, таких як текст, числа, зображення, відео тощо.

Маніпуляційні механізми: Це функції та операції, які дозволяють додавати, змінювати та видаляти дані в базі даних. Наприклад, SQL (Structured Query Language) використовується для виконання операцій з базою даних.

Структура даних: Це спосіб організації даних у базі даних. Вона включає таблиці, поля, зв'язки та інші елементи, які визначають спосіб зберігання та організацію даних.

Запити (Queries): Це запити, які виконуються для отримання певних

даних з бази даних. Запити можуть використовуватися для вибірки даних, оновлення, видалення або з'єднання різних таблиць.

Безпека: Це механізми, які забезпечують захист даних від несанкціонованого доступу та змін. Вона може включати автентифікацію користувачів, авторизацію доступу до даних, шифрування тощо.

Бази даних можуть бути реляційними, об'єктно-орієнтованими, графовими та іншими типами, залежно від структури та вимог конкретного проекту. Вони є ключовим елементом для зберігання та управління інформацією в сучасних інформаційних системах.

Індекси та оптимізація: Індекси допомагають швидко знаходити дані у базі даних, особливо великих обсягів. Їх використання дозволяє зменшити час пошуку і підвищити продуктивність запитів.

Транзакції: Транзакції забезпечують атомарність, консистентність, ізольованість та довіреність (ACID) для змін у базі даних. Це важливо для забезпечення цілісності даних та уникнення проблем у випадку збоїв.

Реплікація та резервне копіювання: Ці механізми дозволяють створювати резервні копії бази даних та реплікувати її для забезпечення надійності та доступності даних у випадку аварій.

Інтеграція з програмним забезпеченням: Бази даних можуть бути інтегровані з іншим програмним забезпеченням, таким як веб-сайти, CRM-системи, ERP-системи та інші, для обміну даними та підтримки різних бізнес-процесів.

Масштабованість: Сучасні бази даних повинні бути здатні масштабуватися для роботи з великими обсягами даних та високими навантаженнями.

Всі ці компоненти та функції баз даних спільно створюють потужний інструмент для зберігання, управління та аналізу даних, який є необхідним у багатьох аспектах сучасного бізнесу та інформаційних технологій.

Для швидкої, зручної та ефективної роботи з базами даних створені

СУБД. СУБД(Система управління баазами даних) — набір взаємопов'язаних даних (база даних) і програм для доступу до цих даних. Надає можливості створення, збереження, оновлення та пошуку інформації в базах даних з контролем доступу до даних.

Модель "сутність-зв'язок" (Entity-Relationship Model, ER-модель) - це концептуальна модель даних, яка використовується для опису структури даних в базі даних. Вона базується на концепціях сутностей, атрибутів та зв'язків між ними.

Основні складові моделі "сутність-зв'язок":

Сутності (Entities): Сутності представляють об'єкти, про які збирається інформація. Наприклад, "Користувач", "Продукт", "Замовлення" - це всі сутності в системі. Кожна сутність має свої атрибути, які описують характеристики цієї сутності.

Атрибути (Attributes): Атрибути - це характеристики сутностей. Наприклад, у сутності "Користувач" атрибутами можуть бути ім'я, прізвище, адреса тощо.

Зв'язки (Relationships): Зв'язки вказують на взаємозв'язки між різними сутностями. Наприклад, в базі даних для інтернет-магазину може бути зв'язок між сутностями "Користувач" і "Замовлення", що показує, який користувач зробив певне замовлення.

Модель "сутність-зв'язок" допомагає відобразити взаємозв'язки між різними сутностями та описати структуру даних у більш зрозумілому та логічному способі. Вона часто використовується на етапі проектування баз даних для створення концептуального опису системи та визначення основних компонентів та їх зв'язків.

Ключі (Keys): В моделі "сутність-зв'язок" ключі використовуються для унікальної ідентифікації кожного запису в базі даних. Один або кілька атрибутів можуть бути обрані як ключі для кожної сутності.

Кардинальність (Cardinality): Кардинальність вказує на кількість екземплярів сутностей, які можуть брати участь у відношенні. Вона може бути один до одного (1:1), один до багатьох (1), багато до одного (M:1) або багато до багатьох (M), де M та N - це числа, що визначають кількість екземплярів.

Нормалізація: Це процес організації даних в базі даних для забезпечення їхньої ефективності та цілісності. Нормалізація допомагає уникнути аномалій даних та забезпечує, що кожна сутність має правильну структуру.

Діаграми ER: Це графічні зображення моделі "сутність-зв'язок", які дозволяють візуально представити структуру даних та їхні взаємозв'язки. Діаграми ER часто використовуються під час аналізу та проектування баз даних.

Модель "сутність-зв'язок" є потужним інструментом для аналізу та проектування баз даних, оскільки вона дозволяє логічно описати структуру даних та їхні взаємозв'язки. Вона є важливою складовою при створенні надійних та ефективних систем зберігання та управління інформацією.

Рисунок 2.4 - ER-діаграма бази даних

ER-діаграма (сутність-відношення) бази даних - це графічне зображення структури бази даних, яке використовується для моделювання відносин між сутностями (об'єктами) в системі та їх атрибутами. Ця діаграма допомагає відобразити сутності (такі як користувачі, продукти, замовлення тощо), їх відносини та атрибути, що характеризують ці сутності.

Основні елементи ER-діаграми включають:

Сутності (Entities): Це об'єкти або поняття, які визначаються як окремі, ідентифіковані об'єкти в системі, які мають свої атрибути. Наприклад, сутність "користувач" може мати атрибути "ім'я", "прізвище", "адреса" і т.д.

Відношення (Relationships): Це зв'язки між сутностями, які показують, як вони пов'язані між собою. Наприклад, зв'язок "замовлення" може вказувати на те, що кожен користувач може мати декілька замовлень, а кожне замовлення може бути пов'язане з одним користувачем.

Атрибути (Attributes): Це властивості або характеристики сутностей, які визначаються для кожної сутності. Наприклад, атрибути для сутності "користувач" можуть включати ім'я, прізвище, електронну пошту тощо.

Ключі (Keys): Це атрибути, які однозначно ідентифікують кожен запис в сутності. Ключі допомагають унікально ідентифікувати кожен запис та забезпечують цілісність даних.

ER-діаграми можуть бути складними, з багатьма сутностями, атрибутами та зв'язками, але вони надають зручний спосіб візуального представлення структури даних в базі даних. Вони допомагають аналізувати вимоги до даних, розробляти моделі даних та забезпечувати консенсус між розробниками та замовниками проекту щодо структури даних.

2.2.4 Опис структури таблиць

Структура таблиці в базі даних може бути описана наступним чином:

Назва таблиці: Це унікальне ідентифікатор для таблиці, яке часто використовується для посилання на неї в запитах SQL.

Стовпці (поля): Кожен стовпець визначає певну характеристику даних, які зберігаються в таблиці. Кожен стовпець має ім'я, тип даних та, за потреби, інші властивості, такі як обмеження на значення або можливість встановлення значення за замовчуванням.

Типи даних: Це формат, в якому дані зберігаються в кожному стовпці. Наприклад, типи даних можуть включати цілі числа, десяткові числа, рядки, дати тощо.

Первинний ключ (Primary Key): Це один або кілька стовпців, які унікально ідентифікують кожен запис в таблиці. Первинний ключ використовується для гарантії унікальності та для посилання на записи в інших таблицях (зв'язки).

Індекси: Індекси використовуються для швидкого доступу до даних у таблиці. Вони створюються для певних стовпців і дозволяють базі даних швидше знаходити рядки з певними значеннями в цих стовпцях.

Зовнішні ключі (Foreign Keys): Це стовпці, які посилаються на первинний ключ іншої таблиці. Вони використовуються для встановлення зв'язків між таблицями.

Обмеження (Constraints): Це правила, які обмежують допустимі значення в таблиці або взаємозв'язки між таблицями. Наприклад, обмеження може забороняти введення негативних значень у числовий стовпець або вимагати, щоб певні поля не були порожніми.

Інші властивості (за потреби): Вони можуть включати такі речі, як захист від змін (імутабельність), тригери (автоматичні дії, які виконуються при вставці, оновленні або видаленні даних) та інше.

Відношення (Relationships): Це взаємозв'язки між таблицями в базі даних. Вони можуть бути один до одного (one-to-one), один до багатьох (one-to-many) або багато до багатьох (many-to-many). Відношення встановлюються за допомогою зовнішніх ключів і дозволяють зв'язувати дані між різними таблицями для ефективного доступу та операцій.

Нормалізація: Це процес організації даних у таблицях таким чином, щоб уникнути аномалій та забезпечити ефективне використання простору та ресурсів бази даних. Нормалізація включає розбиття таблиць на менші та більш узагальнені, що допомагає уникнути повторення даних та забезпечити їх консистентність.

Резервне копіювання та відновлення: Це важливий аспект управління базою даних, що включає створення резервних копій даних для запобігання втраті інформації у випадку аварій або випадкового видалення. Відновлення даних з резервних копій забезпечує збереження даних та бізнес-континуїтет.

Безпека даних: Це набір заходів для захисту даних в базі даних від несанкціонованого доступу, модифікації та видалення. Вона включає в себе контроль доступу, шифрування даних, аудит та інші заходи для забезпечення конфіденційності, цілісності та доступності даних.

Ця структура дозволяє базі даних ефективно зберігати та управляти даними, забезпечуючи їх цілісність та доступність для виконання різноманітних операцій.

Розглянемо структуру бази даних для заданої предметної області. Основними сутностями цієї структури є:

Учасник – описує дані про учасників змагань;

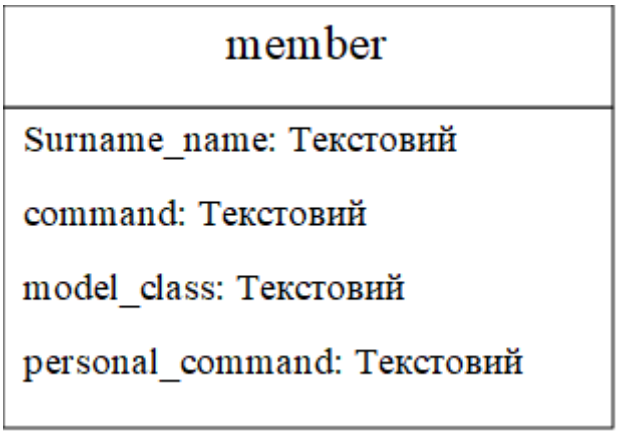
Стендове оцінювання – описує дані про стендові бали учасників;

Ходове оцінювання – описує дані про ходове оцінювання.

Команда – описує дані про команду.

Наступні моделі відображають структуру бази даних:

2.5 – Структура таблиці member



В таблиці member, яка описана на рисунку 2.5, описані наступні поля:

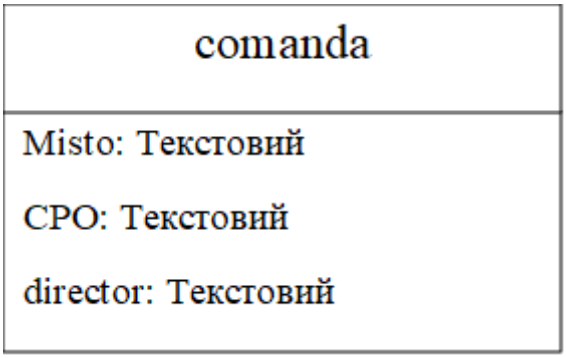
1.surname_name – зберігає прізвище та ім’я учасника;

2. command – зберігає команду, від якої виступає учасник;

3. model_class – зберігає клас моделі з якою виступає учасник;

4. personal_command – зберігає залік учасника (особистий чи командний);

Рисунок 2.6 – Структура таблиці comanda



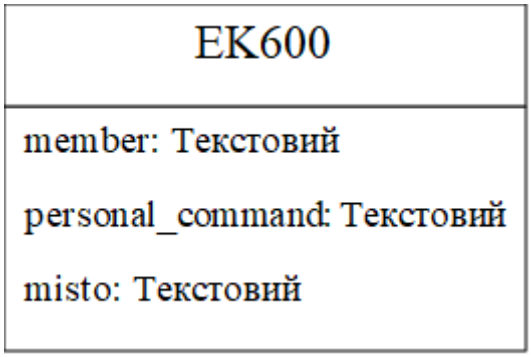
В таблиці comanda, яка описана на рисунку 2.6, описані наступні поля:

Misto – зберігає місто, від якого виступає команда;

CPO – зберігає центр позашкільної освіти;

director – зберігає керівника команди;

Рисунок 2.7 – Структура таблиці EK600



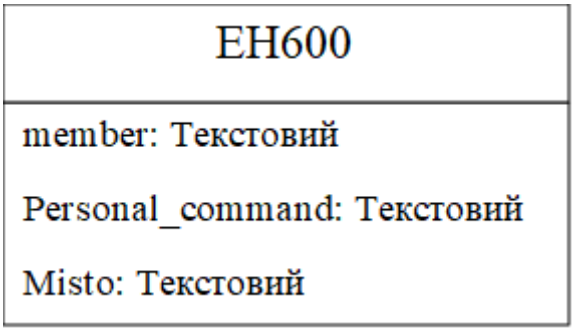
В таблиці EK600, яка описана на рисунку 2.7, описані наступні поля:

member – зберігає учасників, які виступають у даному класі;

personal_command – зберігає залік учасника;

misto – зберігає місто (команду);

Рисунок 2.8 – Структура таблиці EH600



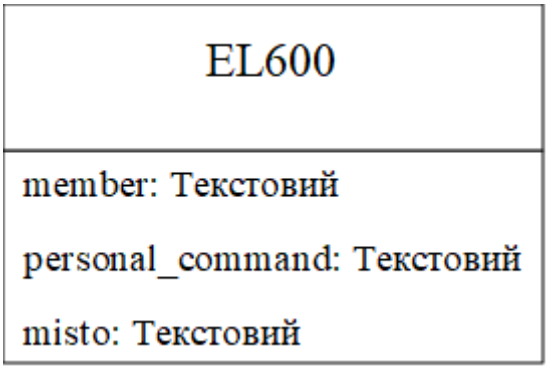
В таблиці EH600, яка описана на рисунку 2.8, описані наступні поля:

member – зберігає учасників, які виступають у даному класі;

personal_command – зберігає залік учасника;

misto – зберігає місто (команду).

Рисунок 2.9 – Структура таблиці EL600



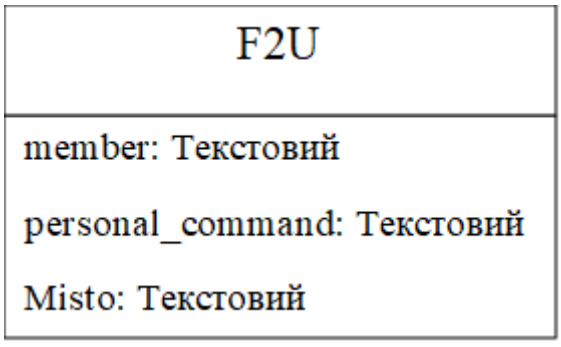
В таблиці EL600, яка описана на рисунку 2.9, описані наступні поля:

member – зберігає учасників, які виступають у даному класі;

Personal_command – зберігає залік учасника;

Misto – зберігає місто (команду).

Рисунок 2.10 – Структура таблиці F2U



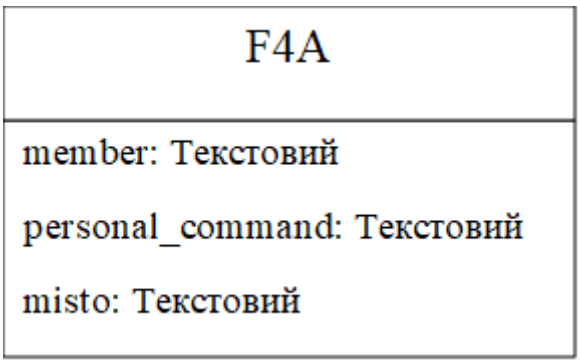
В таблиці F2U, яка описана на рисунку 2.10, описані наступні поля:

member – зберігає учасників, які виступають у даному класі;

personal_command – зберігає залік учасника;

misto – зберігає місто (команду).

Рисунок 2.11 – Структура таблиці F4A



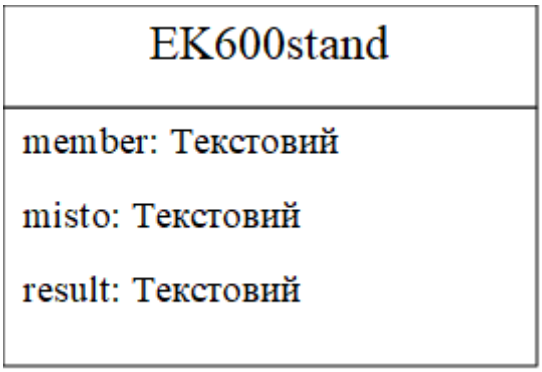
В таблиці F4A, яка описана на рисунку 2.11, описані наступні поля:

member – зберігає учасників, які виступають у даному класі;

personal_command – зберігає залік учасника;

misto – зберігає місто (команду).

Рисунок 2.12 – Структура таблиці EK600stand



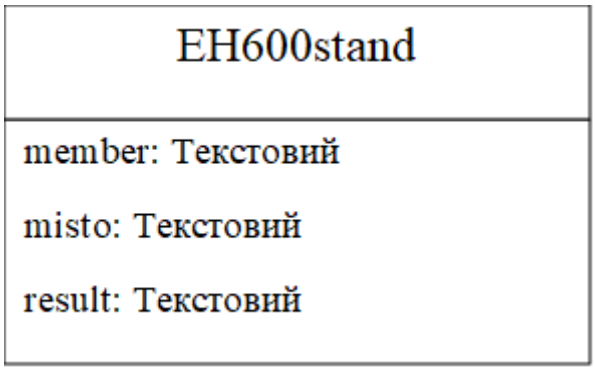
В таблиці EK600stand, яка описана на рисунку 2.12, описані наступні поля:

member – зберігає учасників, які виступають в даному класі;

misto – зберігає місто (команду);

result – зберігає результат стендового оцінювання.

Рисунок 2.13 – Структура таблиці EH600stand



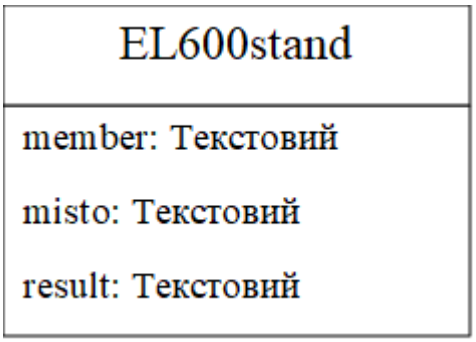
В таблиці EH600stand, яка описана на рисунку 2.13, описані наступні поля:

member – зберігає учасників, які виступають в даному класі;

misto – зберігає місто (команду);

result – зберігає результат стендового оцінювання.

Рисунок 2.14 – Структура таблиці EL600stand



В таблиці EL600stand, яка описана на рисунку 2.14, описані наступні поля:

member – зберігає учасників, які виступають у даному класі;

misto – зберігає місто (команду);

result – зберігає результат стендового оцінювання.

Рисунок 2.15 – Структура таблиці F2Ustand

В таблиці F2Ustand, яка описана на рисунку 2.15, описані наступні поля:

member – зберігає учасників, які виступають у даному класі;

misto – зберігає місто (команду);

result – зберігає результат стендового оцінювання.

2.3. Програмне забезпечення системи

Програмне забезпечення системи відіграє ключову роль у забезпеченні її функціональності та продуктивності. Воно включає в себе різноманітні компоненти, які працюють разом для забезпечення різних функцій та послуг системи. Ось деякі з найважливіших видів програмного забезпечення системи:

Операційна система: Це основне програмне забезпечення, яке керує апаратним забезпеченням і надає середовище для виконання інших програм. Приклади таких операційних систем включають Windows, macOS, Linux та інші.

Програми для обробки даних: Це програмне забезпечення для зберігання, обробки та аналізу даних системи. Це може бути Система управління базами даних (СУБД), така як MySQL, PostgreSQL, Oracle або MongoDB, а також програми для аналізу даних, включаючи Excel, R, Python тощо.

Веб-сервери та веб-додатки: Це програмне забезпечення, яке надає можливість взаємодії з користувачем через Інтернет. Воно включає веб-

сервери, такі як Apache, Nginx, а також веб-додатки, які надають функціональність та послуги через веб-інтерфейс.

Клієнтські програми: Це програмне забезпечення, яке встановлюється на комп'ютери або мобільні пристрої користувачів і дозволяє їм взаємодіяти з системою. Це можуть бути десктопні додатки, мобільні додатки або веб-браузери.

Системи управління версіями: Це програмне забезпечення для керування версіями коду програм та спільної роботи над ними. Приклади таких систем включають Git, Subversion, Mercurial тощо.

Системи аналітики та звітності: Це програмне забезпечення для аналізу даних, створення звітів та візуалізації результатів. Приклади таких систем включають Tableau, Power BI, Google Analytics тощо.

Системи автоматизації бізнес-процесів (BPM): Це програмне забезпечення для автоматизації та управління бізнес-процесами в організації. Приклади включають Microsoft Power Automate, IBM Business Process Manager, Appian тощо.

Системи керування взаємодією з клієнтами (CRM): Це програмне забезпечення для управління відносинами з клієнтами, включаючи обробку замовлень, зв'язок з клієнтами та аналітику. Приклади включають Salesforce, HubSpot CRM, Zoho CRM тощо.

Системи управління вмістом (CMS): Це програмне забезпечення, яке дозволяє організаціям керувати та оновлювати вміст на своєму веб-сайті. CMS дозволяють легко додавати, редагувати та видаляти сторінки, додавати мультимедійний вміст та керувати структурою сайту. Приклади CMS включають WordPress, Drupal, Joomla тощо.

Системи управління проектами (PM): Це програмне забезпечення, яке допомагає командам керувати проектами, розподіляти завдання, встановлювати терміни, відстежувати прогрес та комунікувати. PM системи дозволяють ефективно керувати ресурсами та забезпечити вчасне

завершення проектів. Приклади PM систем включають Trello, Asana, Jira, Microsoft Project тощо.

Системи управління контентом (ECM): Це програмне забезпечення, яке дозволяє організаціям керувати, зберігати, організовувати та забезпечувати доступ до різних видів контенту, включаючи документи, веб-контент, електронну пошту тощо. ECM дозволяють забезпечити ефективне управління інформацією та дотримуватися регулятивних вимог. Приклади ECM систем включають SharePoint, Documentum, Alfresco тощо.

Системи управління виробництвом (ERP): Це програмне забезпечення, яке допомагає організаціям керувати всіма аспектами свого бізнесу, включаючи фінанси, закупівлі, виробництво, логістику, продажі та багато іншого. ERP системи дозволяють підприємствам оптимізувати свою діяльність та забезпечувати її ефективне управління. Приклади ERP систем включають SAP, Oracle ERP, Microsoft Dynamics тощо.

Системи управління взаємодією зі споживачами (CIM): Це програмне забезпечення, яке допомагає організаціям встановлювати та підтримувати взаємодію зі споживачами через різні канали комунікації, включаючи соціальні медіа, електронну пошту, чати та інші. CIM системи дозволяють підприємствам підтримувати ефективні взаємодії зі споживачами та підвищувати рівень задоволеності клієнтів. Приклади CIM систем включають Salesforce Service Cloud, Zendesk, Intercom тощо.

Ці різні види програмного забезпечення працюють разом для забезпечення різних аспектів функціональності та послуг системи. Вибір відповідних програмних засобів залежить від конкретних потреб та вимог системи.

2.3.1. Системне програмне забезпечення

Системне програмне забезпечення - це набір програм, які забезпечують функціональність та ресурси для роботи комп'ютерної системи. Ось декілька типових компонентів системного програмного забезпечення:

Операційна система (ОС): Це основний компонент системного програмного забезпечення, який керує ресурсами комп'ютера та надає середовище для виконання програм. Операційна система контролює процеси, пам'ять, введення-виведення та інші аспекти комп'ютерної системи.

Драйвери пристроїв: Ці програми забезпечують взаємодію між операційною системою та конкретними пристроями або периферійними пристроями, такими як принтери, сканери, графічні карти тощо. Вони дозволяють ОС коректно керувати та взаємодіяти з пристроями.

Утиліти системного адміністрування: Це програми, які допомагають адміністраторам керувати та налаштовувати різні аспекти операційної системи та комп'ютерної мережі. Це може включати утиліти для моніторингу ресурсів, резервного копіювання, відновлення системи тощо.

Бібліотеки програмного забезпечення: Це набори функцій та процедур, які можуть бути використані розробниками програм для спрощення розробки. Бібліотеки можуть включати функції для роботи з мережами, обробки даних, графікою тощо.

Віртуальні машини та емулятори: Це програмне забезпечення, яке надає можливість запускати віртуальні операційні системи або емулювати апаратне забезпечення, що дозволяє виконувати програми, специфічні для інших операційних систем або апаратних платформ, на поточній системі.

Системні бібліотеки: Це набори функцій та процедур, які забезпечують основні можливості програмного забезпечення на рівні операційної системи. Це може включати роботу з файлами, мережеві операції, керування пам'яттю та інші системні операції.

Системне програмне забезпечення для мереж: Це програмне забезпечення, яке забезпечує керування мережевими ресурсами та забезпечує зв'язок між комп'ютерами в мережі. До такого програмного

забезпечення можуть входити сервери доменних імен (DNS), сервери файлів, маршрутизатори, мережеві файрволи тощо.

Системне програмне забезпечення для безпеки: Це програмне забезпечення, яке забезпечує захист комп'ютерної системи від зловмисних атак, вірусів, шпигунського програмного забезпечення та інших загроз безпеці. До такого програмного забезпечення можуть входити антивіруси, антишпигунське програмне забезпечення, файрволи, системи виявлення вторгнень (IDS), системи керування доступом тощо.

Системне програмне забезпечення для віддаленого керування: Це програмне забезпечення, яке дозволяє адміністраторам керувати комп'ютерами або серверами віддалено, необхідне для управління мережами та серверами на відстані. До такого програмного забезпечення можуть входити програмні інструменти для віддаленого доступу, системи керування конфігурацією, системи моніторингу та інші інструменти.

Системне програмне забезпечення для віртуалізації: Це програмне забезпечення, яке дозволяє створювати та керувати віртуальними об'єктами, такими як віртуальні машини або віртуальні приватні сервери (VPS). Віртуалізація дозволяє ефективно використовувати апаратні ресурси та управляти різними середовищами на одному фізичному сервері.

Системне програмне забезпечення для управління базами даних (СУБД): Це програмне забезпечення, яке забезпечує зберігання, доступ, пошук та обробку даних в базах даних. Воно включає в себе такі компоненти, як системи керування базами даних (СКБД), які дозволяють створювати, модифікувати та оптимізувати структуру баз даних, а також мови запитів для отримання даних з бази.

Системне програмне забезпечення для розробки інтегрованих систем: Це програмне забезпечення, яке дозволяє розробникам створювати програми для взаємодії з різними компонентами системи, такими як апаратне забезпечення, операційна система, мережеві ресурси тощо. Це може

включати в себе середовища розробки, SDK (набори розробника), інструменти для тестування та налагодження, API та інші компоненти.

Системне програмне забезпечення для керування ресурсами: Це програмне забезпечення, яке дозволяє керувати та оптимізувати використання ресурсів комп'ютерної системи, таких як процесор, пам'ять, мережеві ресурси, сховища даних тощо. Воно може включати утиліти моніторингу, системи планування завдань, системи управління пам'яттю та інші компоненти.

Системне програмне забезпечення для системного аналізу та моделювання: Це програмне забезпечення, яке допомагає аналізувати, проектувати та моделювати різні аспекти комп'ютерної системи, такі як архітектура, продуктивність, безпека тощо. Воно може включати інструменти для розробки діаграм, симуляцій та інших методів аналізу та моделювання.

Ці компоненти разом утворюють інфраструктуру, необхідну для ефективної роботи комп'ютерних систем і додатків.

Розроблений програмний застосунок призначений для встановлення на персональному комп’ютері користувача.

Таблиця 2.1 Сумісність ОС

Операційна система	
Microsoft Windows	Windows 10
	Windows 8.1
	Windows 8
	Windows 7

3 РОЗРОБКА ДЕСКТОП-ДОДАТКУ

3.1. Опис програмних модулів системи

Опис програмних модулів системи допомагає користувачам краще розуміти функції та можливості кожного модуля. Він також служить як

довідковий матеріал для розробників та адміністраторів системи. Ось кілька кроків, які можна включити до опису програмних модулів:

Назва модуля: Почніть з назви модуля, щоб ідентифікувати його унікально.

Огляд функцій: Надайте короткий огляд функцій, які надає модуль, та його цільового призначення.

Опис інтерфейсу користувача: Поясніть, як користувачі можуть взаємодіяти з модулем через інтерфейс користувача. Включіть опис доступних опцій, кнопок, полів введення тощо.

Опис функцій та операцій: Детально опишіть функції та операції, які можна виконати за допомогою модуля. Включіть крок-за-кроком інструкції та приклади використання.

Залежності та інтеграції: Зазначте залежності модуля від інших компонентів системи, а також можливість інтеграції з іншими програмними модулями або системами.

Безпека та доступність: Опишіть заходи безпеки, які використовуються для захисту модуля від несанкціонованого доступу, а також рівень доступності та надійності модуля.

Параметри конфігурації та налаштування: Поясніть, які параметри можна налаштувати для модуля, і як це можна зробити.

Ліцензія та власник: Вкажіть тип ліцензії, яка регулює використання модуля, а також власника або розробника модуля.

Приклади використання: Надайте декілька прикладів використання модуля для кращого розуміння його можливостей та потенціалу.

Підтримка та контактна інформація: Зазначте контактні дані для отримання підтримки, якщо виникнуть питання чи проблеми з використанням модуля.

Ці елементи допоможуть створити повний та зрозумілий опис кожного програмного модуля системи, який буде корисним для кінцевих користувачів та розробників.

Програмний застосунок представляє собою сукупність форм, кожна з яких відповідає за певний набір функцій.

MainForm – Форма, призначена для переходу на інші форми. Приклад форми изображено на малюнку 2.16.

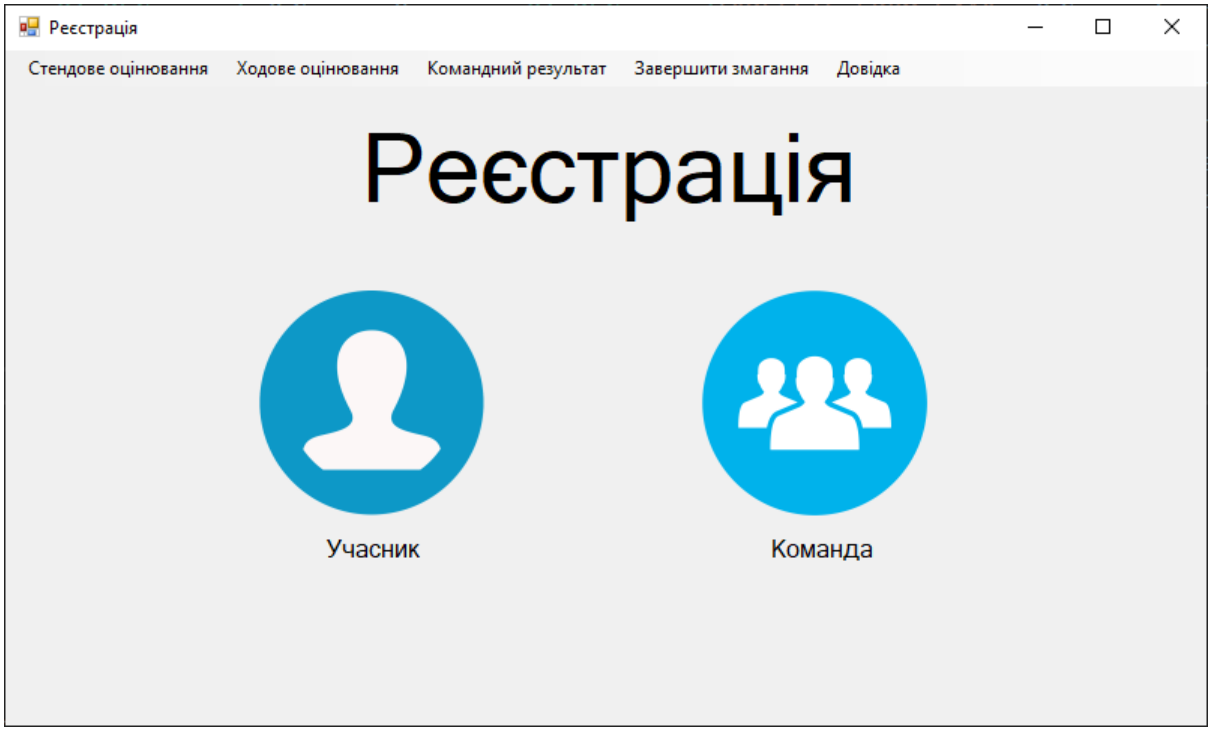


Рисунок 3.1 – Форма реєстрації

Register_Command – Форма, призначена для реєстрації команди. Приклад зображено на малюнку 3.2.

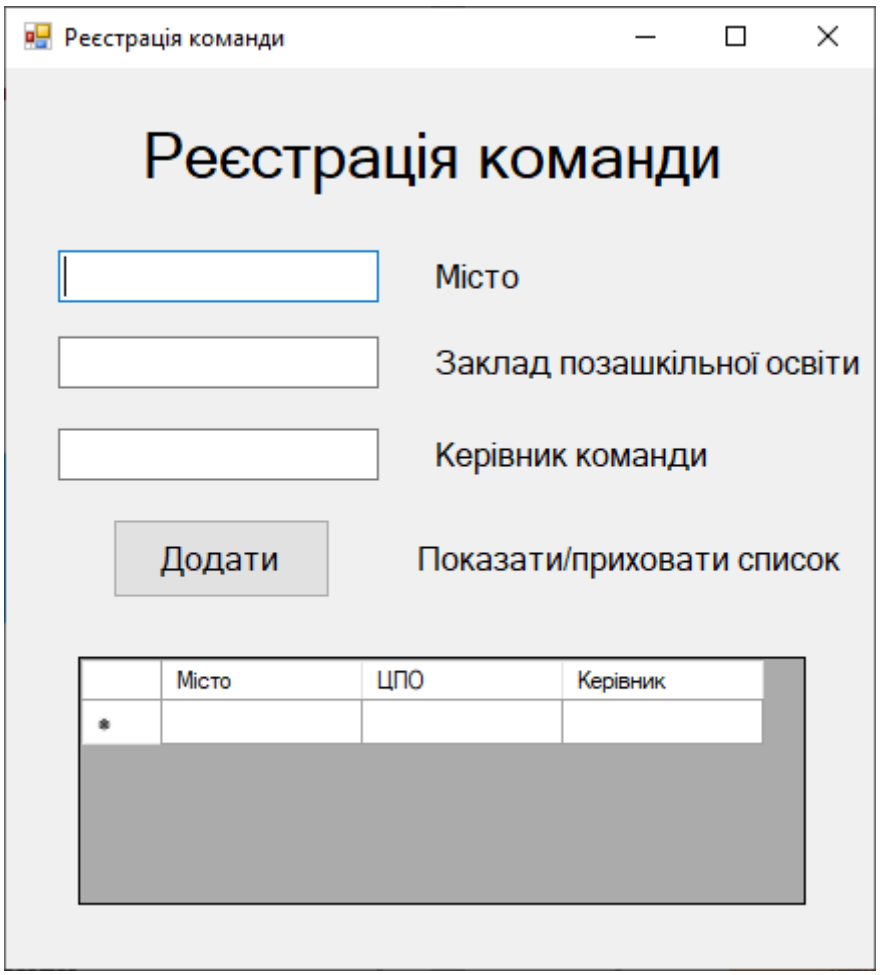


Рисунок 3.2 – Форма реєстрації команди

Register_member – Форма, призначена для реєстрації учасників змагань. Приклад зображено на рисунку 3.3.

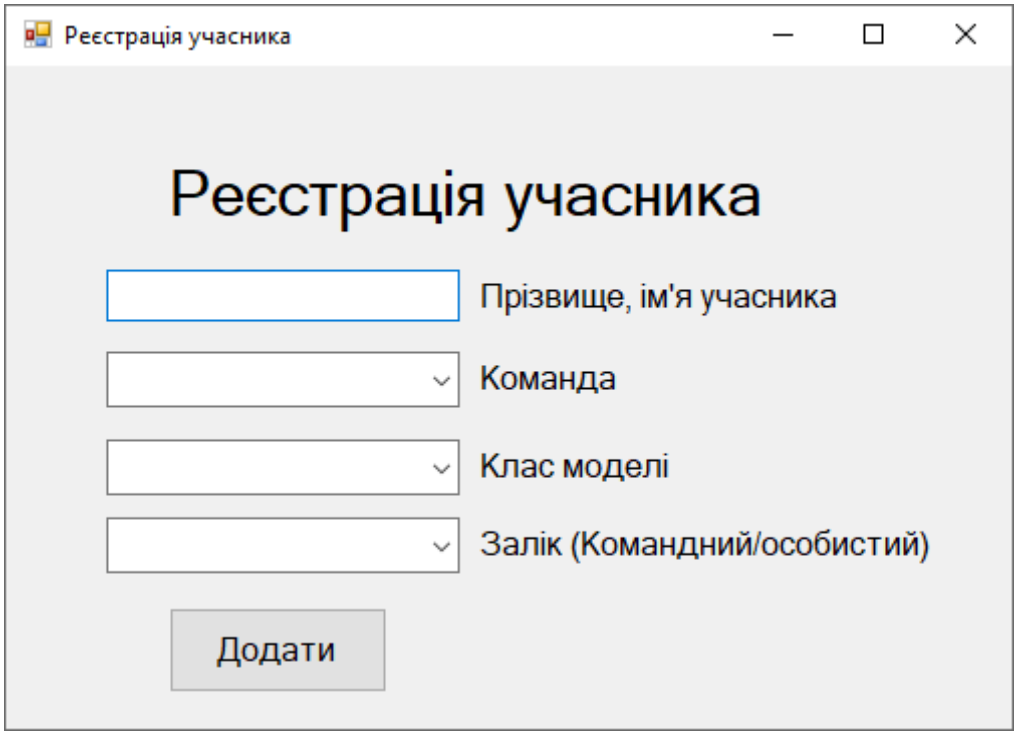
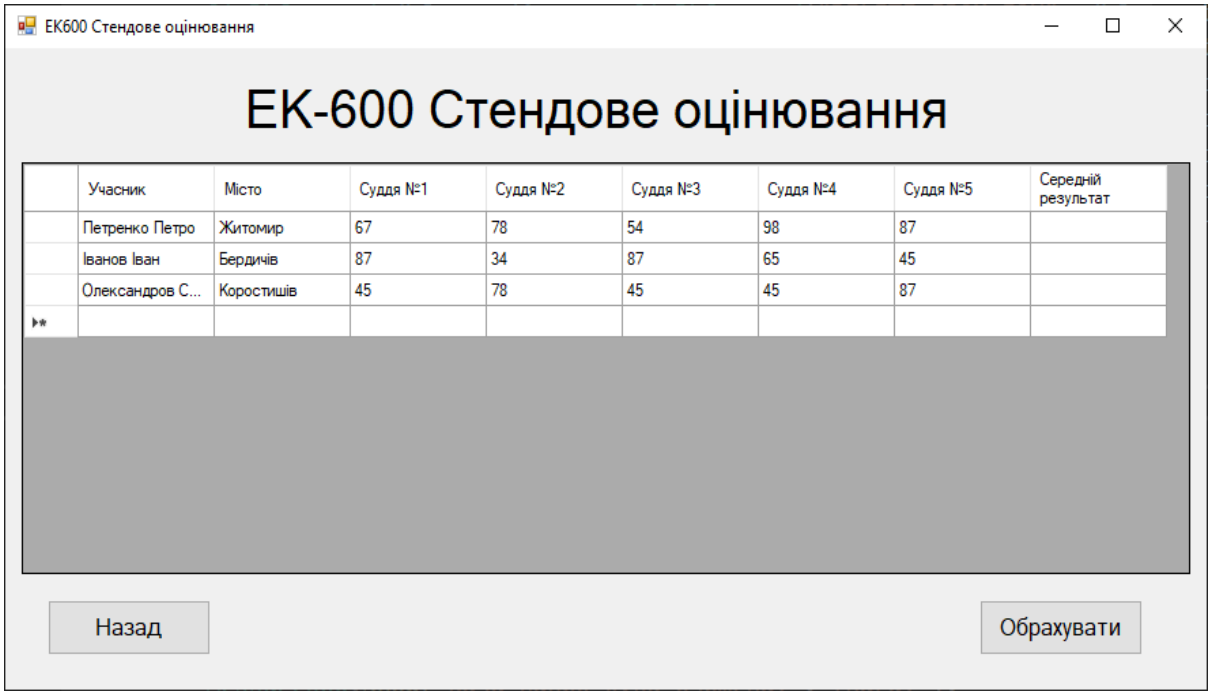


Рисунок 3.3 – Форма реєстрації учасників

ЕК600 – Форма, призначена для введення і обрахунку балів за стендове оцінювання для класу моделей ЕК600. Приклад зображено на рисунку 3.4.



	Учасник	Місто	Суддя №1	Суддя №2	Суддя №3	Суддя №4	Суддя №5	Середній результат
	Петренко Петро	Житомир	67	78	54	98	87	
	Іванов Іван	Бердичів	87	34	87	65	45	
	Олександров С...	Коростишів	45	78	45	45	87	
▸*								

Рисунок 3.4 – Форма стендового оцінювання класу ЕК600

ЕН600 – Форма, призначена для введення і обрахунку балів за стендове оцінювання моделей класу ЕН600. Приклад зображено на малюнку 3.5.

EK600 -Масштабні моделі військових(бойових) кораблів довжиною до 600 мм

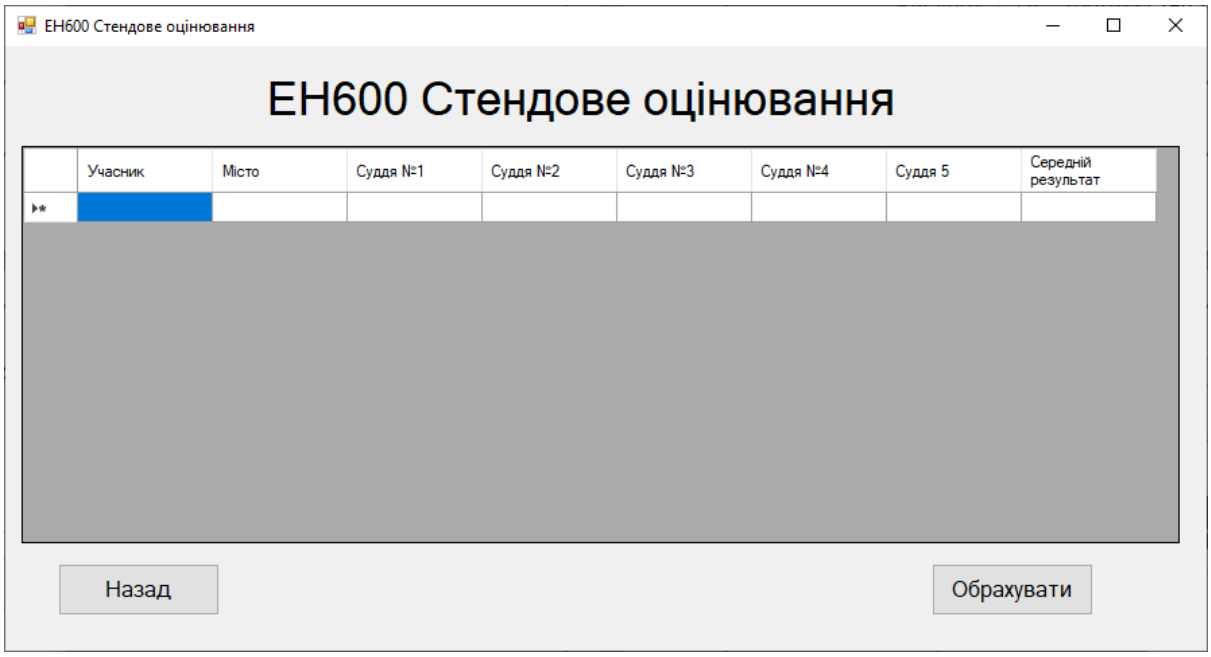


Рисунок 3.5 – Форма стендового оцінювання класу EH600

EL600 – Форма, призначена для введення і обрахунку балів за стендове оцінювання моделей класу EL600. Приклад зображено на малюнку 3.6.

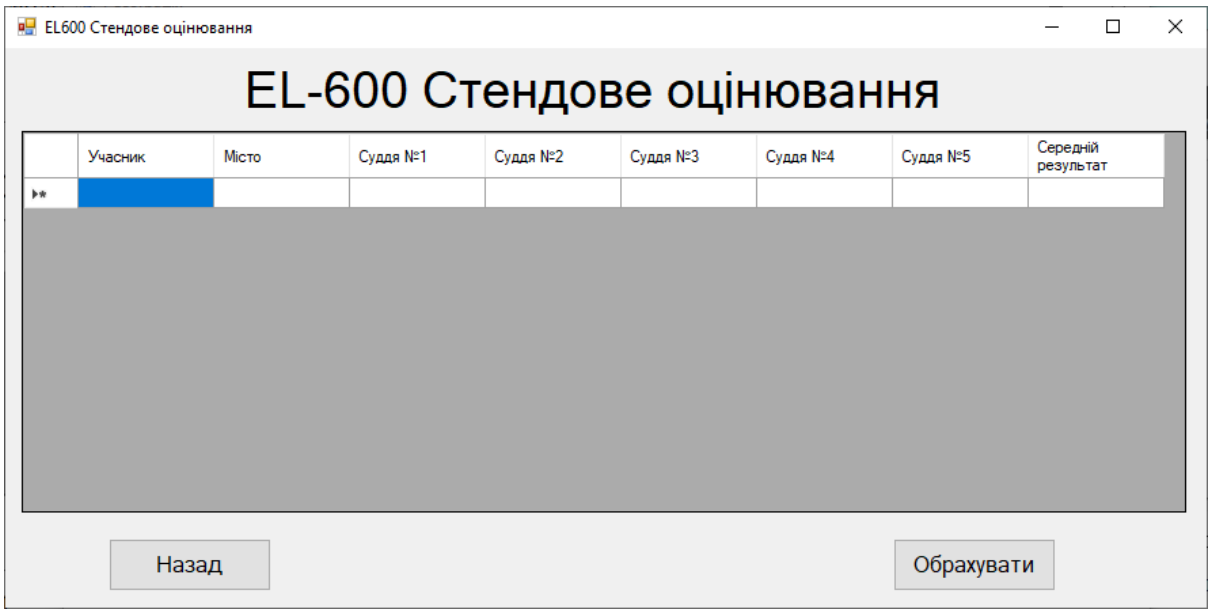


Рисунок 3.6 – Форма стендового оцінювання класу EL600

F2U – Форма, призначена для введення і обрахунку балів за стендове оцінювання моделей класу F2U. Приклад зображено на рисунку 3.7.

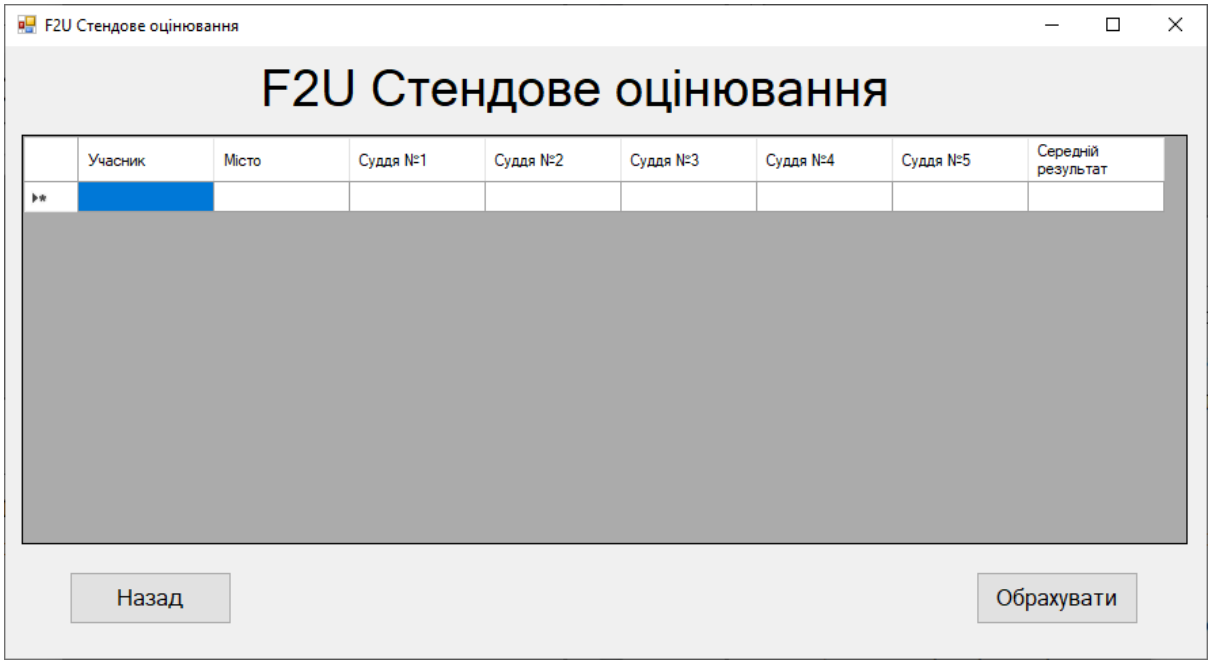


Рисунок 3.7 – Форма стендового оцінювання класу F2U

EK600move – форма, призначена для введення і обрахунку балів за ходове оцінювання класу EK600. Приклад зображено на рисунку 3.8

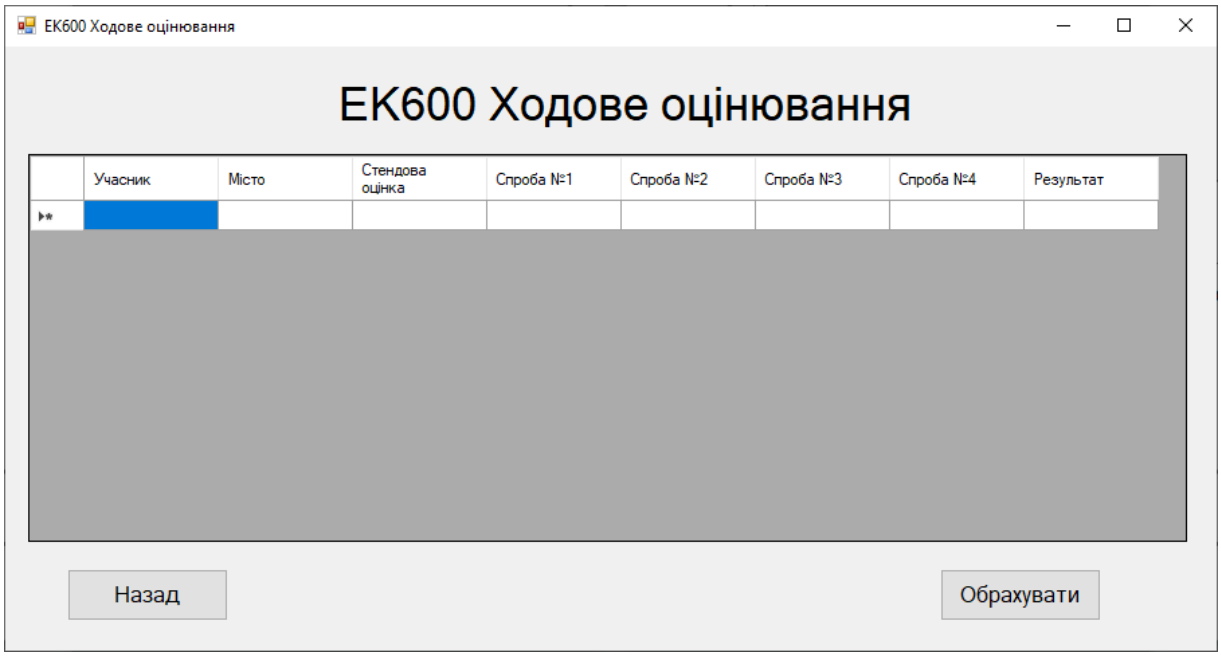


Рисунок 3.8 – Форма ходового оцінювання класу ЕК600

ЕН600move – форма, призначена для введення і обрахунку балів за ходове оцінювання класу ЕН600. Приклад зображено на рисунку 3.9.

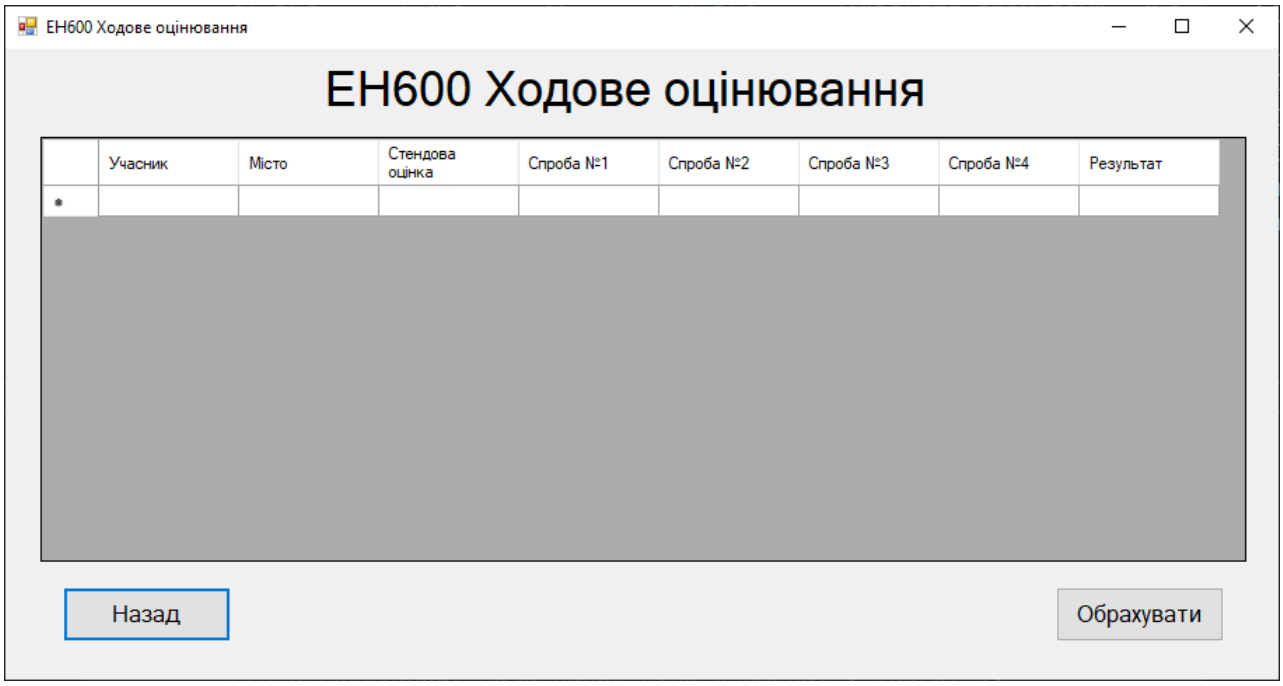


Рисунок 3,9 – Форма ходового оцінювання класу ЕН600

EL600move – форма, призначена для введення і обрахунку балів за ходове оцінювання класу EL600. Приклад зображено на рисунку 3.10

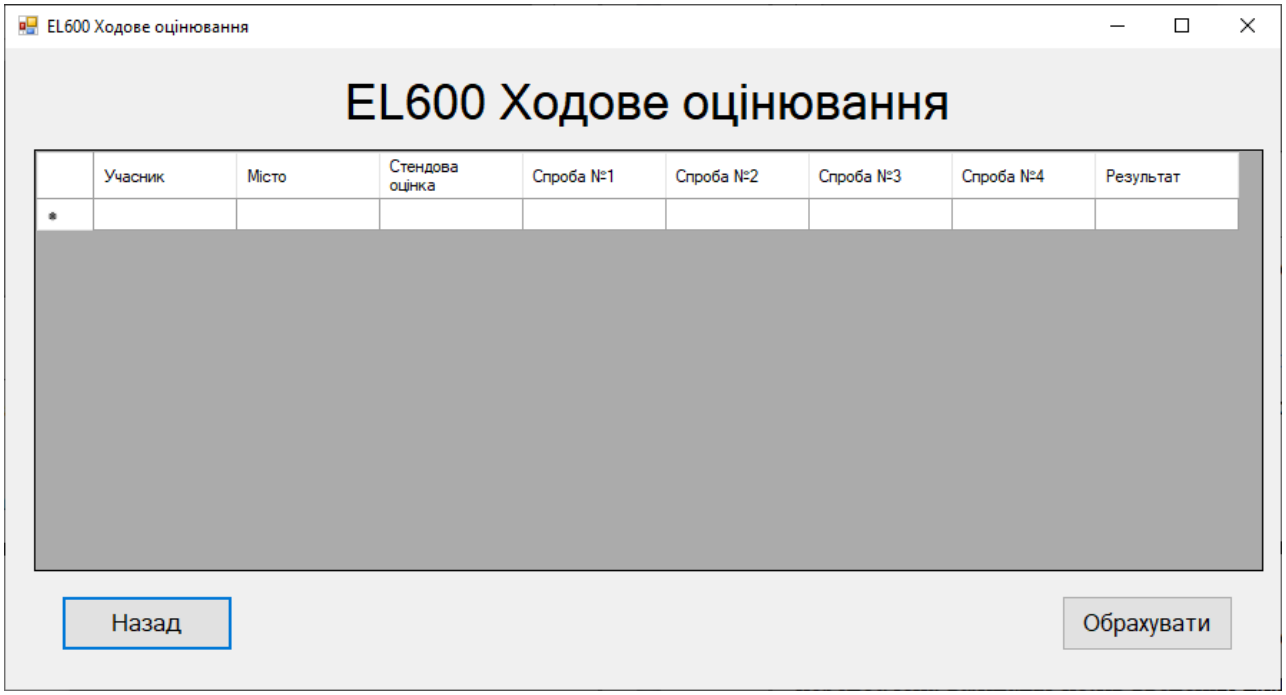


Рисунок 3.10 – Форма ходового оцінювання класу EL600

F2Umove – форма, призначена для введення і обрахунку балів за ходове оцінювання класу F2U. Приклад зображено на рисунку 3.11.

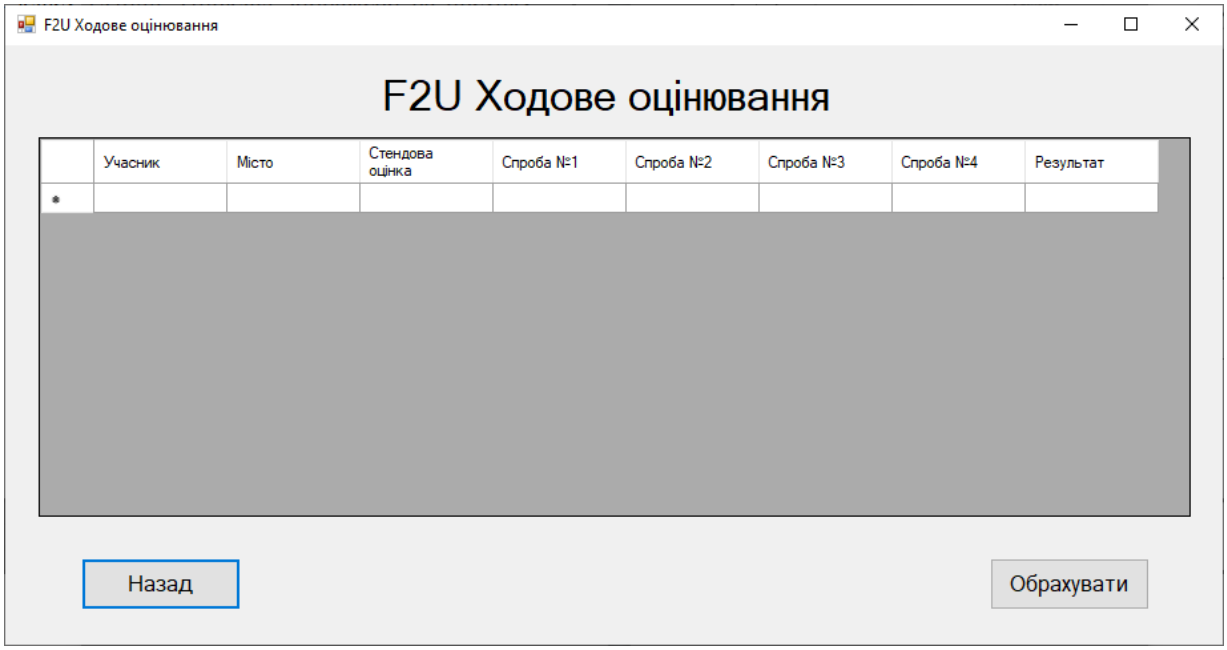


Рисунок 3.11 – Форма ходового оцінювання класу F2U

F4Amove – Форма, призначена для введення і обрахунку балів за ходове оцінювання класу F4A. Приклад зображено на рисунку 3.12.

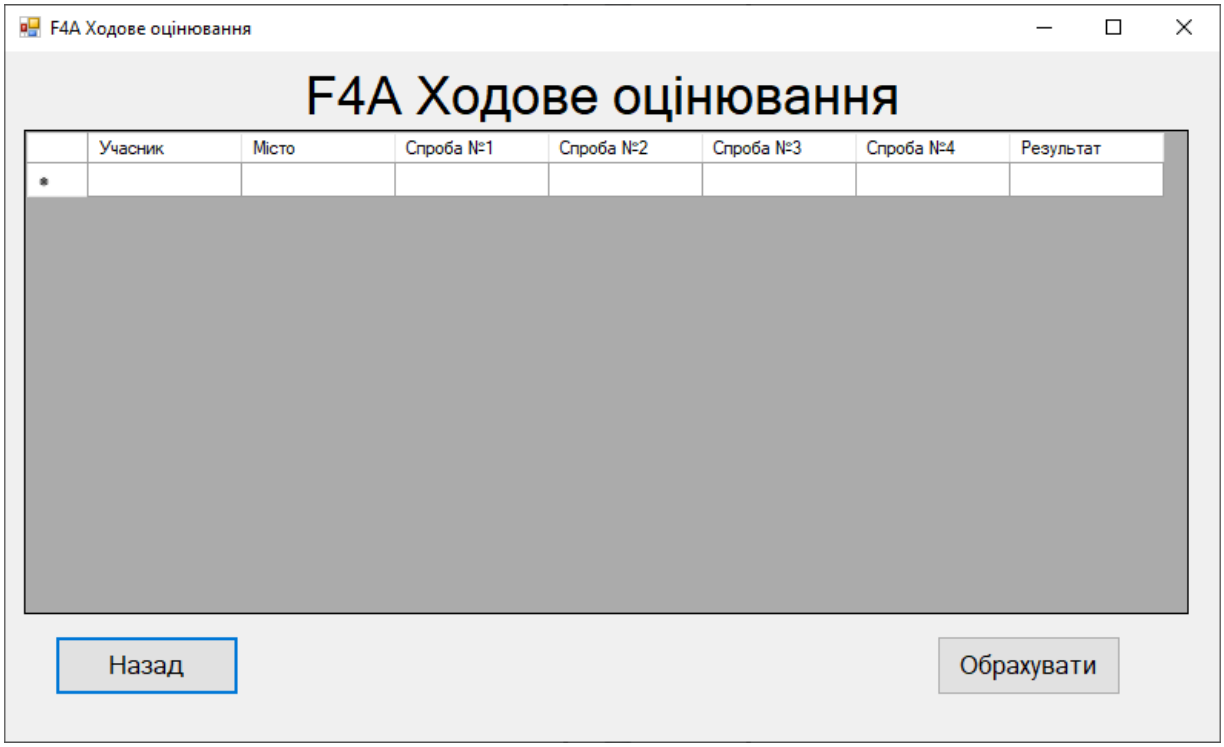


Рисунок 3.12 – Форма ходового оцінювання класу F4A

Command_result – форма, призначена для визначення командних результатів. Приклад зображено на рисунку 3.14

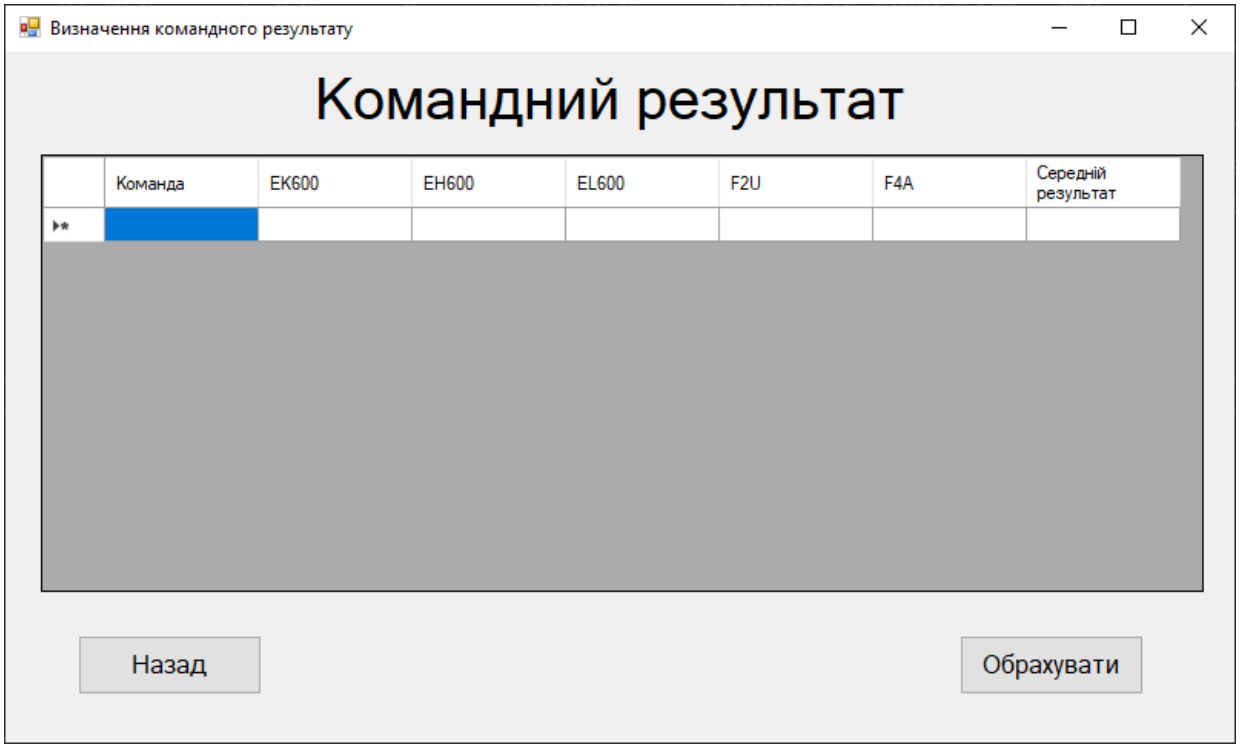


Рисунок 3.14 – Форма визначення командних результатів

Рішення щодо забезпечення надійної роботи системи

Для забезпечення надійності роботи програми було виключено усі можливі способи нестабільної роботи програми. Програма працює без збоїв.

Оцінка можливих причин помилок та збоїв у роботі системи та способів запобігання їх виникнення

Під час тестування роботи програми помилок та збоїв виявлено не було. Якщо такі виникнуть, слід звернутися до розробника за підтримкою.

3.2. Технічне забезпечення системи

Технічне забезпечення системи включає в себе всі аспекти апаратного та програмного забезпечення, необхідного для її нормального функціонування. Ось деякі ключові компоненти технічного забезпечення системи:

Апаратне забезпечення: Це обладнання, яке використовується для роботи системи, таке як сервери, комп'ютери, маршрутизатори, комутатори тощо. Важливо, щоб апаратне забезпечення було потужним та надійним для забезпечення високої продуктивності та доступності системи.

Операційна система: Це базове програмне забезпечення, яке керує апаратним забезпеченням і надає середовище для виконання інших програм. Вибір операційної системи залежить від потреб системи та вимог до її функціональності.

Бази даних: Це програмне забезпечення для зберігання та управління даними системи. Важливо вибрати відповідну базу даних, яка забезпечить швидкий доступ до даних, високу надійність та безпеку.

Мережеве забезпечення: Це інфраструктура для забезпечення зв'язку між різними компонентами системи, включаючи мережеве обладнання, протоколи зв'язку та програмне забезпечення для мережевого управління.

Захист та безпека: Це програмне та апаратне забезпечення для захисту системи від зловмисних атак, вірусів, витоку конфіденційної інформації та інших загроз безпеці. Включає в себе антивірусне програмне забезпечення, файерволи, інструменти моніторингу безпеки тощо.

Резервне копіювання та відновлення: Це процедури та технології для регулярного створення резервних копій даних системи та їх відновлення в разі аварій.

Моніторинг та управління: Це програмне забезпечення для моніторингу стану системи, аналізу продуктивності, виявлення проблем та віддаленого управління.

нтеграція систем: Деякі системи можуть потребувати інтеграції з іншими програмами або системами для обміну даними або спільного використання ресурсів. Це може включати в себе розробку та реалізацію спеціальних інтерфейсів та протоколів комунікації.

Інфраструктура хмарних обчислень: Використання хмарних послуг для забезпечення обчислювальних, мережевих та сховищевих потреб системи. Це може включати в себе використання хмарних платформ, таких як Amazon Web Services (AWS), Microsoft Azure або Google Cloud Platform (GCP).

Масштабованість: Забезпечення можливості розширення або зменшення масштабів системи в залежності від змін обсягу даних або завдань. Це може включати в себе використання горизонтального або вертикального масштабування, а також розробку та впровадження систем автоматичного масштабування.

Управління версіями: Забезпечення контролю версій програмного забезпечення та його компонентів, щоб забезпечити правильне впровадження оновлень та вирішення проблем безпеки.

Доступність та надійність: Розробка системи з урахуванням вимог до доступності та надійності, включаючи використання резервних джерел

живлення, дублювання компонентів системи та розробку планів відновлення після аварій.

Швидкість та продуктивність: Забезпечення ефективної роботи системи та швидкого відгуку на запити користувачів шляхом оптимізації алгоритмів, використання кешування даних та інших технік оптимізації продуктивності.

Забезпечення відповідності до вимог законодавства та стандартів безпеки: Дотримання вимог щодо захисту даних, конфіденційності, доступності та інших аспектів, встановлених законодавством або стандартами галузі.

Ці компоненти разом складають технічне забезпечення системи і допомагають забезпечити її стабільну та ефективну роботу. Важливо обирати та налаштовувати ці компоненти з урахуванням конкретних потреб та вимог системи.

3.2.1. Обґрунтування вибору технічного забезпечення системи

Для розробки програмного проекту був використаний персональний комп’ютер з технічними характеристиками, які наведені у таблиці 2.2.

Таблиця 3.2.1 Технічні характеристики обладнання

Комплектуючі	Модель, основні характеристики обладнання
Процесор	Intel Core I3-9100F 3.6GHz
Оперативна пам'ять	8 ГБ (DDR4-2400)
Жорсткий диск	128 Гб SSD + 1 ТБ HDD
Екран	22” (1920x1080)
Графічний адаптер	Nvidia GTX 1050 Ti 4GB
Операційна система	Windows 10

3.3. Методичне забезпечення системи

Методичне забезпечення системи — це набір документів, матеріалів, рекомендацій та інших ресурсів, які сприяють ефективній організації та функціонуванню системи. Це може включати усе, від навчальних посібників і методичних посібників для працівників до стандартів та процедур, що регулюють діяльність системи.

Методичне забезпечення важливе для забезпечення якості та стабільності системи. Ось кілька ключових аспектів, які можуть бути включені до нього:

Навчальні матеріали: Посібники, які навчають користувачів або працівників, як користуватися системою.

Стандарти і процедури: Документи, які визначають норми та правила для виконання різних завдань в системі.

Документація системи: Повний опис функціональності системи, її складових частин, архітектури, конфігурації та інших технічних аспектів.

Посібники користувача: Інструкції та поради для кінцевих користувачів щодо використання системи.

Методичні матеріали для підтримки: Інформаційні ресурси для підтримки та вирішення проблем, які можуть виникнути під час експлуатації системи.

Оновлення та покращення: Процедури для впровадження змін у систему, включаючи оновлення програмного забезпечення та вдосконалення функціональності.

Тестові матеріали: Методики тестування системи та перевірки її працездатності.

Безпека і конфіденційність: Матеріали, що стосуються заходів безпеки та політик конфіденційності, які повинні дотримуватися при роботі з системою.

Технічна підтримка: Документація та процедури щодо отримання технічної підтримки для вирішення технічних проблем, які можуть виникнути.

Тренінги та семінари: Організація навчальних заходів для користувачів системи з метою поглиблення їхніх знань та навичок щодо її використання.

Моніторинг та оцінка: Методи та інструменти для вимірювання ефективності та продуктивності системи, а також збору зворотного зв'язку від користувачів для подальших поліпшень.

Документування процесів: Створення документів, які описують процеси роботи з системою, включаючи процедури реєстрації, обробки заявок, вирішення проблем тощо.

Внутрішні комунікації: Засоби та процедури для ефективної комунікації між працівниками, які використовують систему, для обміну інформацією та кращого співробітництва.

Дослідження та розвиток: Програми для постійного вдосконалення системи на основі оцінки нових технологій, потреб користувачів та інших факторів.

Ці складові доповнюють та підтримують основні аспекти методичного забезпечення системи, створюючи комплексний підхід до її управління та ефективного використання.

Забезпечення цієї системи може вимагати постійного оновлення та адаптації до нових вимог і викликів, але наявність чіткого та комплексного методичного забезпечення допомагає забезпечити ефективне використання системи та зменшити ймовірність помилок.

3.3.1. Методика тестування системи

Тестування програмного забезпечення - процес дослідження, випробування програмного продукту, який має на меті перевірку

відповідності між реальною поведінкою програми і її очікуваним поведінкою на кінцевому наборі тестів, обраних певним чином.

У різний час і в різних джерелах тестування давалися різні визначення, в тому числі:

процес виконання програми з метою знаходження помилок;

інтелектуальна дисципліна, що має на меті одержання надійного програмного забезпечення без зайвих зусиль на його перевірку;

технічне дослідження програми для отримання інформації про її якість з точки зору певного кола зацікавлених осіб.

перевірка відповідності між реальною поведінкою програми і її очікуваним поведінкою на кінцевому наборі тестів, виконаних певним чином;

процес спостереження за виконанням програми в спеціальних умовах і винесення на цій основі оцінки будь-яких аспектів її роботи;

процес, який має на меті виявлення ситуацій, в яких поведінка програми є неправильним, небажаним або не відповідає специфікації;

процес, що містить у собі всі активності життєвого циклу, як динамічні, так і статичні, що стосуються планування, підготовки та оцінки програмного продукту і пов'язаних з цим результатів робіт з метою визначити, що вони відповідають описаним вимогам, показати, що вони підходять для заявлених цілей і для визначення дефектів.

Методика впровадження та розгортання системи

Для зручного встановлення програмного застосунку на комп'ютер користувача було розроблено інсталятор. Приклад зображено на малюнку 3.3.1. За домомогою інсталятора можна зручно обирати потрібну папку для встановлення. Інсталятор був розроблений за допомогою розширення Smart Install Maker.

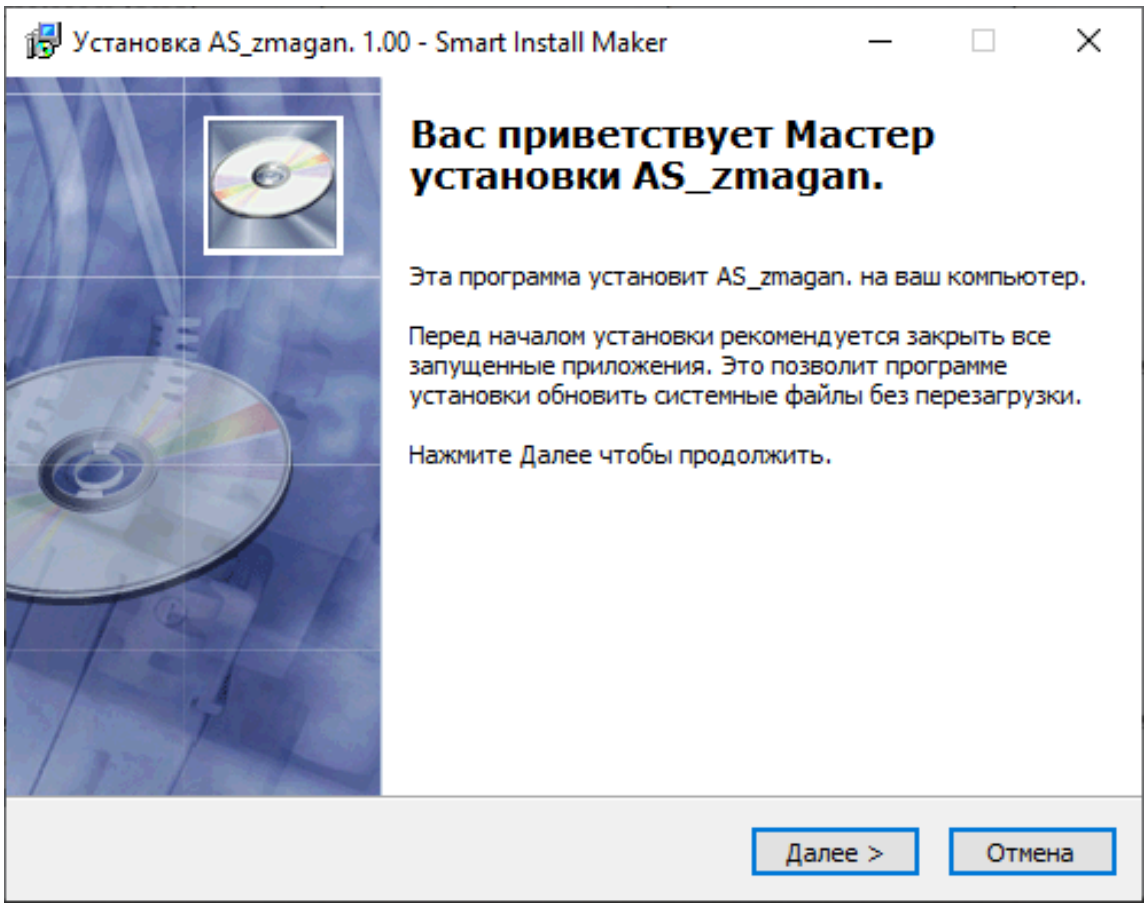


Рисунок 3.3.1 – Приклад зовнішнього вигляду інсталятора

3.4. Інструкція користувача

Інструкція користувача є важливим документом, який надає користувачам необхідну інформацію для успішного використання системи або програмного забезпечення. Ось кілька кроків, які можна включити до інструкції користувача:

Вступ: Зазначте загальну мету та призначення системи або програмного забезпечення, а також короткий опис функцій, які вона надає.

Вимоги до системи: Вкажіть мінімальні вимоги до апаратного та програмного забезпечення для використання системи.

Процес реєстрації/входу в систему: Поясніть, як користувачі можуть зареєструватися у системі або увійти до неї.

Огляд інтерфейсу користувача: Надайте огляд інтерфейсу користувача та поясніть, як користуватися основними елементами інтерфейсу.

Основні функції та операції: Поясніть основні функції системи та операції, які користувачі можуть виконувати.

Кроки для виконання певних завдань: Надайте детальні інструкції з виконання певних завдань або операцій у системі, включаючи крок-за-кроком інструкції та скріншоти, якщо це можливо.

Підказки та поради: Надайте корисні поради та підказки для полегшення використання системи та підвищення продуктивності користувачів.

Часті питання (FAQ): Включіть перелік часто задаваних питань та їх відповідей для вирішення загальних проблем або непорозумінь користувачів.

Контактна інформація для підтримки: Зазначте контактні дані для отримання допомоги або підтримки в разі виникнення проблем.

Заключні вказівки: Надайте короткі вказівки або поради щодо безпеки, збереження даних та інші важливі аспекти використання системи

Ці кроки допоможуть створити повну та зрозумілу інструкцію користувача, яка сприятиме успішному використанню системи або програмного забезпечення.

Оновлення та нові функції: Поясніть, як користувачі можуть отримувати оновлення системи або програмного забезпечення, а також інформуйте їх про нові функції та можливості, які доступні в нових версіях.

Правила безпеки та конфіденційності: Подайте короткий огляд правил безпеки та конфіденційності, які користувачам слід дотримуватися під час використання системи або програмного забезпечення.

Словник термінів: Включіть словник термінів з поясненням ключових термінів та скорочень, що використовуються у системі або програмному забезпеченні.

Приклади використання: Надайте короткі приклади використання системи або програмного забезпечення для кращого розуміння його потенціалу та можливостей.

Налаштування та персоналізація: Поясніть, як користувачі можуть налаштовувати та персоналізувати систему або програмне забезпечення згідно зі своїми потребами та вподобаннями.

Довідкові матеріали: Надайте посилання на додаткові довідкові матеріали, такі як відео-інструкції, посібники користувача або онлайн-ресурси, які допоможуть користувачам краще зрозуміти систему або програмне забезпечення.

Загальні рекомендації та поради: Підкресліть загальні рекомендації та поради щодо ефективного використання системи або програмного забезпечення, такі як регулярне збереження даних, резервне копіювання тощо.

Довідкова система була розроблена у вигляді HTML файлу, виклик якого здійснюється безпосередньо з програми. Відкривається файл довідки у браузері за замовчуванням. Приклад довідкової системи зображено на малюнку 3.4.1

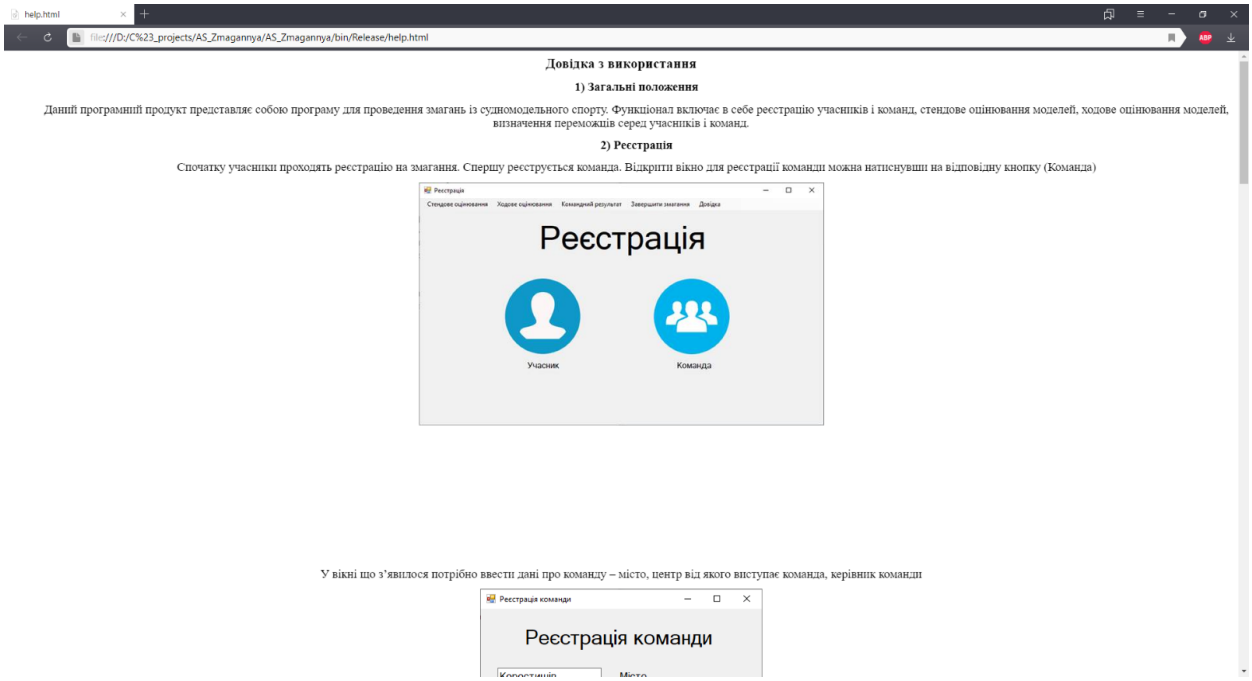


Рисунок 3.4.1 – Приклад вікна довідкової системи

4 БІЗНЕС-ПЛАН

4.1 Опис продукту

4.1.1 Вступ

Розробка десктопної програми для автоматизованого проведення змагань з судомодельного спорту може мати декілька вагомих переваг:

Ефективність організації змагань: Завдяки десктопній програмі можна ефективно організувати різні етапи змагань, такі як реєстрація учасників, формування розкладу змагань, проведення жеребкування, підрахунок результатів тощо. Це дозволяє зменшити час, необхідний для підготовки та проведення змагань, та підвищити їхню організаційну ефективність.

Точність та надійність обробки даних: Використання програмного забезпечення дозволяє уникнути помилок та неточностей при обробці результатів змагань. Система автоматизованого підрахунку результатів може забезпечити надійність та точність результатів, що важливо для справедливого розподілу місць та нагород між учасниками.

Підвищення привабливості змагань для учасників та глядачів: Завдяки використанню сучасних технологій, таких як візуалізація результатів на екрані, інтерактивні таблиці лідерів, онлайн трансляції тощо, можна зробити процес проведення змагань більш захоплюючим і цікавим для учасників та глядачів.

Можливості аналізу та статистики: Десктопна програма може забезпечити можливості для збору та аналізу статистичних даних про учасників та результати змагань. Це дозволяє організаторам отримати більше інформації про характеристики змагань, виявити тенденції та покращити їхню організацію в майбутньому.

Зручність для користувачів: Використання десктопної програми спрощує взаємодію з організаторами змагань для учасників, судей та інших учасників. Вони можуть швидко отримувати доступ до розкладу змагань,

своїх результатів, а також зв'язуватися з організаторами через систему сповіщень або чату.

Система реєстрації учасників та управління даними: Десктопна програма може включати зручну систему реєстрації учасників, де вони можуть заповнювати свої дані та реєструватися на змагання. Крім того, програма може забезпечити централізоване управління даними учасників, їхніми результатами та іншою інформацією, що дозволяє зберігати та оновлювати дані ефективно.

Система автоматизованого жеребкування та складання розкладу змагань: Програма може автоматично проводити жеребкування для розподілу учасників на кваліфікаційні групи або змагання та складати розклад змагань з урахуванням різних факторів, таких як категорії суден, попередні результати тощо.

4.1.2 Основні функції та переваги

Система пропонує широкий спектр функцій, що дозволяють користувачам ефективно керувати своїми фінансами:

Ефективність організації змагань: Програма дозволить значно зменшити час, необхідний для підготовки та проведення змагань. Всі аспекти такі як реєстрація учасників, управління розкладом, реєстрація результатів і журі можуть бути автоматизовані, що дозволяє зменшити витрати на організацію події.

Зниження витрат: Замість традиційних паперових форматів і ручного оброблення даних, програма дозволить значно зекономити на витратах на папір, друку та ручній роботі.

Покращення якості обслуговування: Автоматизація дозволяє запобігти помилкам та знизити ризик виникнення конфліктних ситуацій під час проведення змагань, що веде до покращення задоволення учасників та глядачів.

Розвиток інфраструктури: Створення такої програми може стимулювати розвиток інфраструктури судомодельного спорту, оскільки полегшує його організацію та зростання популярності.

Можливість комерційного використання: Якщо програма буде розроблена з урахуванням потреб комерційних організаторів змагань, вона може стати предметом продажу або використання на платній основі, що відкриває можливості для генерації прибутку.

Загалом, розробка десктопної програми для автоматизації змагань з судомодельного спорту може принести значні користі в економічному плані, сприяючи ефективнішому використанню ресурсів та покращенню управлінських процесів в цій галузі.

4.1.3 Цільова аудиторія

Цільова аудиторія включає широкий спектр користувачів, кожен з яких має свої специфічні потреби у використанні системи:

Організатори змагань: Люди або організації, які відповідають за планування, організацію та проведення судомодельних змагань. Вони використовуватимуть програму для автоматизації реєстрації учасників, управління розкладом, обробки результатів та інших адміністративних завдань.

Учасники змагань: Люди, які беруть участь у судомодельних змаганнях. Для них програма може надавати можливість реєстрації, доступу до інформації про змагання, розкладу, результатах та іншої важливої інформації.

Судді та журі: Люди, які відповідають за оцінку та організацію змагань. Для них програма може надавати зручний інтерфейс для введення та оцінки результатів учасників, а також для комунікації з іншими членами журі.

Глядачі та фанати: Люди, які відвідують змагання або слідкують за ними в онлайн-режимі. Деякі програмні рішення можуть надавати можливості для онлайн-трансляцій змагань, оновлення результатів у реальному часі та інші функції, що залучають глядачів.

Тренери та команди: У випадку командних змагань, тренери та члени команди також можуть користуватися програмою для координації своїх учасників, моніторингу результатів та аналізу даних.

Ця програма спрямована на полегшення організаційних аспектів судомодельних змагань для всіх вищезгаданих груп користувачів, забезпечуючи ефективність, точність та зручність у проведенні та участі в таких заходах.

4.2 Аналіз ринку

4.2.1 Огляд ринку

Ринок програмного забезпечення для автоматизованого проведення змагань з судомодельного спорту є швидко зростаючим. Зростаюча фінансова грамотність та необхідність контролю витрат сприяють зростанню попиту на такі інструменти. Десктоп-додатки для проведення змагань з судомодельного спорту стають дедалі популярнішими завдяки своїй доступності, функціональності та зручності використання.

Потреба на ринку: Змагання з судомодельного спорту часто потребують складної організації та точного управління, включаючи реєстрацію учасників, управління розкладом, обробку результатів і взаємодію з журі. Для більшої ефективності та точності управління подіями сучасні організатори часто шукають рішення у вигляді програмного забезпечення.

Конкуренція: На ринку існує кілька гравців, які пропонують різні рішення для автоматизації спортивних змагань. Деякі з них спеціалізуються

на загальних рішеннях для різних видів спорту, тоді як інші можуть концентруватися саме на судомодельному спорті.

4.2.2 Оцінка попиту

Попит на десктопні програми для автоматизованого проведення змагань з судомодельного спорту є значним і базується на ряді факторів, які включають зростання популярності спорту, потребу в ефективному управлінні змаганнями та тренди на цифровізацію в спорті загалом.

Зростання популярності судомодельного спорту: Зацікавлення у судомодельному спорті зростає серед учасників, тренерів та глядачів. Це створює попит на ефективні та професійно організовані змагання, які можуть бути покращені через використання спеціалізованого програмного забезпечення.

Потреба в ефективному управлінні змаганнями: Організатори змагань та клуби потребують засобів для автоматизації процесів, таких як реєстрація учасників, управління розкладом, розрахунок результатів та комунікація з учасниками. Десктопні програми надають зручний і ефективний спосіб керування цими аспектами.

Підвищення якості організації інших аспектів: Використання програм для автоматизації дозволяє знизити ймовірність помилок та покращити точність управлінських процесів, що в свою чергу сприяє підвищенню якості проведення змагань та задоволенню учасників і глядачів.

Потреба у зборі та аналізі даних: Сучасні програми можуть надавати розширені можливості для збору, зберігання та аналізу даних про учасників, результати, статистику та інші параметри, що є важливими для тренування, підготовки до змагань та стратегічного планування.

4.3 Маркетингова стратегія

4.3.1 Ціноутворення

Для монетизації продукту пропонуються наступні моделі ціноутворення:

Безкоштовна версія:

Обмежений функціонал, підтримуваний рекламою.

Дозволяє користувачам ознайомитися з основними можливостями системи без фінансових витрат.

Преміум-підписка:

Повний доступ до всіх функцій без реклами.

Місячна або річна оплата.

Додаткові функції, такі як розширені аналітичні інструменти, інтеграція з банківськими рахунками, персоналізовані рекомендації.

Разовий платіж:

Одноразова покупка без подальших платежів, що надає довічний доступ до преміум-функцій.

Привабливий варіант для користувачів, які не бажають підписуватися на регулярні платежі.

4.3.2 Канали розповсюдження

Основними каналами розповсюдження будуть:

Онлайн-маркетинг:

Використання соціальних мереж (Facebook, Instagram, LinkedIn) для просування продукту та залучення нових користувачів.

SEO-оптимізація для підвищення видимості в пошукових системах.

Контекстна реклама в Google Ads, яка дозволяє залучати користувачів, зацікавлених у фінансових інструментах.

Email-маркетинг для підтримання зв'язку з існуючими користувачами, інформування про нові функції, акції та інші важливі новини.

Контент-маркетинг: створення та публікація корисного контенту на тему фінансового планування, обліку витрат та доходів, що сприятиме залученню нових користувачів та покращенню репутації продукту.

Партнерства:

Співпраця з фінансовими консультантами, банками та іншими фінансовими установами для просування продукту серед їх клієнтів.

Включення продукту до пакетів фінансових послуг, що пропонуються банками та іншими фінансовими установами.

Мобільні платформи:

Розповсюдження через App Store та Google Play, що дозволить залучити користувачів мобільних пристроїв.

Використання платформи Progressive Web Apps (PWA) для забезпечення кросплатформеного доступу до продукту без необхідності завантаження з магазинів додатків.

Реклама на спеціалізованих платформах:

Реклама на фінансових та бізнесових форумах, веб-сайтах та блогах, що мають аудиторію, зацікавлену в управлінні фінансами.

Участь у фінансових виставках та конференціях для просування продукту серед професіоналів та зацікавлених користувачів.

4.4 Фінансовий план

4.4.1 Прогноз доходів

Прогноз доходів від десктопних програм для автоматизованого проведення змагань з судомодельного спорту може бути складним через різноманітність факторів, що впливають на ринок. Однак, для уявлення про можливий дохід, можна розглянути наступні аспекти:

Ціна продажу програми: Зазвичай десктопні програми для організації змагань мають різні цінові моделі, від одноразової покупки до підписки або використання на основі оплати за кожне проведене змагання. Ціна може

варіюватися в залежності від функціональних можливостей та обсягу послуг, що надаються.

Обсяг ринку: Оцінка потенційного обсягу ринку, який включає кількість клубів і асоціацій судомодельного спорту, які можуть зацікавитися використанням таких програм.

Конкуренція: Врахування конкуренції на ринку, що дозволяє приблизно оцінити частку ринку, яку може захопити новий продукт.

Стратегія маркетингу та продажу: Впровадження ефективної стратегії маркетингу та продажу для просування програми серед потенційних клієнтів.

Прогноз кількості продажів: Базований на аналізі ринку і ринкових тенденцій можна зробити прогноз кількості продажів програми протягом певного періоду часу.

4.4.2 Початкові витрати

Початкові витрати на розробку десктопної програми для автоматизованого проведення змагань з судомодельного спорту можуть значно варіюватися в залежності від кількох факторів, включаючи обсяг функціональності, складність програми, кваліфікацію команди розробників та інші технічні та організаційні аспекти. Ось основні складові витрат:

Розробка програмного забезпечення: Це включає оплату розробників, які працюють над створенням програми. Вартість може залежати від кількості розробників, їх кваліфікації та тривалості розробки.

Дизайн інтерфейсу користувача (UI/UX): Важливий аспект для забезпечення зручності і ефективності використання програми користувачами.

Тестування і контроль якості: Витрати на внутрішнє тестування для виявлення помилок та забезпечення відповідності стандартам якості.

Інтеграція та підтримка: Вартість інтеграції програми з іншими системами (якщо потрібно) і підтримка користувачів після випуску.

Ліцензії і сертифікація: Якщо програма має включати спеціалізовані сертифікації або ліцензії для використання у спортивних заходах.

Маркетинг і просування: Витрати на просування програми, включаючи веб-сайт, рекламні матеріали, участь у виставках та інші маркетингові заходи.

4.5 План розвитку

План розвитку передбачає постійне вдосконалення продукту та розширення функціоналу. Основні напрямки розвитку включають:

Впровадження нових функцій:

Додавання нових функцій на основі відгуків користувачів, таких як більш детальна аналітика, додаткові інструменти планування, інтеграція з іншими фінансовими сервісами.

Впровадження автоматичних нагадувань та сповіщень про важливі фінансові події та строки.

Розширення можливостей аналітики та автоматизації:

Розробка більш детальних та інтуїтивно зрозумілих аналітичних інструментів для кращого розуміння фінансових потоків.

Впровадження інструментів для автоматизації повторюваних завдань, таких як автоматичне категоризування витрат, планування платежів тощо.

Інтеграція з новими фінансовими інструментами та сервісами:

Інтеграція з новими банками та фінансовими установами для забезпечення більшого охоплення користувачів.

Впровадження підтримки криптовалют та інших нових фінансових інструментів.

Постійне оновлення безпеки та захисту даних:

Регулярні оновлення системи безпеки для захисту даних користувачів від нових загроз.

Впровадження додаткових заходів безпеки, таких як біометрична автентифікація, моніторинг підозрілої активності тощо.

Розширення на нові ринки:

Вихід на нові регіональні ринки, включаючи локалізацію продукту для різних мов та культурних особливостей.

Співпраця з місцевими фінансовими установами та партнерами для збільшення охоплення.

Зворотний зв'язок та підтримка користувачів:

Постійне покращення служби підтримки користувачів для швидкого вирішення проблем та запитів.

Використання зворотного зв'язку від користувачів для вдосконалення продукту та впровадження нових функцій.

Цей бізнес-план дозволяє побачити перспективи розвитку продукту та забезпечити його успішне впровадження на ринок. Виконання запланованих заходів сприятиме зростанню бази користувачів, збільшенню доходів та зміцненню позицій продукту на ринку.

ВИСНОВОК

У даному дипломному проекті була розроблена база даних та програмний застосунок автоматизованої стистеми визначення рейтингів учасників гуртка технічної творчості. Для побудови цієї системи були використані мова програмування C# та СУБД Access. У якості IDE була використана Microsoft Visual Studio.

Програма має наступний функціонал:

- Створення та редагування кошторисів;
- Створення звітів та їх друк;
- Управління користувачами та різнорівневий доступ;
- Додавання інформації у таблиці довідники;

В роботі представлено аналіз поставленої задачі, вивчено специфіку предметної області. Також опрацьовано теоретичний матеріал, викладено суть та теоретичні основи досліджуваної проблеми, обрано інструментальні засоби, визначенні основні завдання проекту, проведено огляд сучасної літератури, присвяченої питанням, що розглядаються.

Можливе доопрацювання програми у бік покращення інтерфейсу, розробка серверної бази даних таким чином, можливість оперувати однаковими даними з різних комп’ютерів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. dotNet som multimediaplattform https://hj.divaportal.org/smash/record.jsf?pid=diva2%3A24437&dswid=6315
2. ActiveX, and .NET (dotNet) https://www.sciencedirect.com/science/article/abs/pii/B978024080932850012 0?via%3Dihub
3. Implementation of Web Content Extraction of Structured Data Using DotNet Framework https://www.ijcseonline.org/full_paper_view.php?paper_id=2038
4. Research and Analysis of Image Processing Technologies Based on DotNet Framework https://www.sciencedirect.com/science/article/pii/S1875389212007766?via%3Dihub
5. DOTnet 2.0: Deep learning network for diffuse optical tomography image reconstruction https://www.sciencedirect.com/science/article/pii/S2666521223000479?via%3Dihub
6. SQL∗: A recursive SQL https://www.sciencedirect.com/science/article/abs/pii/030643799390009P?via%3Dihub
7. NO SQL- NOT OBLIGATORY SQL (NATURAL LANGUAGE TO SQL CONVERSION) https://www.irjmets.com/uploadedfiles/paper//issue_4_april_2023/35725/final/fin_irjmets1681270055.pdf
8. Die SQL-Norm (The SQL Standard) https://www.degruyter.com/document/doi/10.1524/itit.45.1.30.19031/html
9. SQL multimedia and application packages (SQL/MM) https://dl.acm.org/doi/10.1145/604264.604280
10. SQL standardization https://dl.acm.org/doi/10.1145/344788.344819
11. Entity Framework https://link.springer.com/chapter/10.1007/978-1-4302-2456-3_8
12. .NET Framework https://link.springer.com/referenceworkentry/10.1007/978-3-319-17885-1_100001

13. Translating Temporal SQL to Nested SQLhttps://digitalcommons.usu.edu/etd/4966/
14. Antidote SQL: SQL for Weakly Consistent Databases https://run.unl.pt/handle/10362/68859
15. SQL on Hops https://kth.divaportal.org/smash/record.jsf?pid=diva2%3A1149002&dswid=631
16. Коноваленко І. В. Програмування мовою C# 6.0. : навчальний посібник для технічних спеціальностей вищих навчальних закладів. Тернопіль ТНУ. 2016. 229 с. URL: https://exam.nuwm.edu.ua/pluginfile.php/103770/mod_resource/content/1/Konovalenko_I_Programuvannja_movoju_C%23_6_0_%282016%29.pdf
17. Основи об’єктно-орієнтованого програмування у C# : методичні вказівки до лабораторних робіт для студентів І-го курсу математичногофакультету спеціальності "Прикладна математика" / Брила А. Ю. та ін.Ужгород, 2014. 73 с. https://exam.nuwm.edu.ua/pluginfile.php/103774/mod_resource/content/1/C%23_%D0%9E%D0%9E%D0%9F.pdf
18. Pro WPF in C# 2010: Windows Presentation Foundation in .NET 4" by Matthew MacDonald https://books.google.com.ua/books?id=nYl7J7z3KssC&printsec=frontcover&redir_esc=y#v=onepage&q&f=false
19. "C# 7.0 in a Nutshell: The Definitive Reference" by Joseph Albahari and Ben Albahari https://kupichitay.com.ua/product/c-7-0-in-a-nutshell-the-definitive-reference-joseph-albahari/
20. "WPF 4.5 Unleashed" by Adam Nathan https://books.google.com.ua/books?id=ubgRAAAAQBAJ&printsec=frontcover&redir_esc=y#v=onepage&q&f=false
21. "Programming WPF" by Chris Sells and Ian Griffiths https://books.google.com.ua/books/about/Programming_WPF.html?id=558i6t1dKEAC&redir_esc=y
22. "C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development" by Mark J. Price https://alek772.github.io/Books/Csharp%208.0%20and%20.NET%20Core%203.0%20%E2%80%93%20Modern%20Cross-Platform%20Development%204th%20Edition.pdf
23. "MVVM in Practice" by Brian Lagunas https://brianlagunas.com/
24. "Pro C# 7: With .NET and .NET Core" by Andrew Troelsen and Philip Japikse

https://dl.ebooksworld.ir/motoman/Apress.Pro.Csharp.7.With.NET.and.NET.Core.www.EBooksWorld.ir.pdf
25. "Practical C# and WPF for Financial Markets" by Jack Xu https://books.google.com.ua/books/about/Practical_C_and_WPF_For_Financial_Market.html?id=qTgRMQAACAAJ&redir_esc=y
26. Windows Presentation Foundation Unleashed (WPF)" by Adam Nathan https://dokumen.pub/wpf-4-unleashed-1nbsped-0672331195-9780672331190.html
27. "WPF Recipes in C# 2008: A Problem-Solution Approach" by Sam Noble, Sam Bourton, and Allen Jones https://link.springer.com/book/10.1007/978-1-4302-1083-2
28. "Microsoft Access 2019 Programming by Example with VBA, XML, and ASP" by Julitta Korol https://terrorgum.com/tfox/books/microsoftaccess2019programmingwithvbaxmlandasp.pdf
29. "Microsoft Access 2016 Programming By Example: with VBA, XML, and ASP" by Julitta Korol https://dokumen.pub/microsoft-access-2016-programming-by-example-with-vba-xml-and-asp-1nbsped-9781944534509.html
30. "Programming Entity Framework: Code First" by Julia Lerman https://www.oreilly.com/library/view/programming-entity-framework/9781449317867/

ДОДАТОК 1

Лістинг програмного коду

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
using System.Diagnostics;

namespace AS_Zmagannya
{
    public partial class Form1 : Form
    {
        public Form1()
        {
```

```
    InitializeComponent();
}

private void button1_Click(object sender, EventArgs e)
{
    Register_member af = new Register_member();
    af.Owner = this;
    af.Show();
}

private void button2_Click(object sender, EventArgs e)
{
    Register_command af = new Register_command();
    af.Owner = this;
    af.Show();
}

private void button3_Click(object sender, EventArgs e)
{

}

private void eK600ToolStripMenuItem_Click(object sender, EventArgs e)
{
    ek600stand af = new ek600stand();
    af.Owner = this;
    af.Show();
}

private void eH600ToolStripMenuItem_Click(object sender, EventArgs e)
{
    EH600stand af = new EH600stand();
    af.Owner = this;
    af.Show();
}

private void eL600ToolStripMenuItem_Click(object sender, EventArgs e)
{
    el600stand af = new el600stand();
    af.Owner = this;
    af.Show();
}

private void f2UToolStripMenuItem_Click(object sender, EventArgs e)
{
    F2Ustand af = new F2Ustand();
    af.Owner = this;
    af.Show();
}

private void f4AToolStripMenuItem_Click(object sender, EventArgs e)
{

}

private void eK600ToolStripMenuItem1_Click(object sender, EventArgs e)
{
    ek600move af = new ek600move();
    af.Owner = this;
    af.Show();
}
```

```
        private void eH600ToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            eh600move af = new eh600move();
            af.Owner = this;
            af.Show();
        }

        private void eL600ToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            el600move af = new el600move();
            af.Owner = this;
            af.Show();
        }

        private void f2UToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            F2Umove af = new F2Umove();
            af.Owner = this;
            af.Show();
        }

        private void f4AToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            F4Amove af = new F4Amove();
            af.Owner = this;
            af.Show();
        }

        private void командний РезультатToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            comand_result af = new comand_result();
            af.Owner = this;
            af.Show();
        }

        private void pictureBox1_Click(object sender, EventArgs e)
        {
            Register_member af = new Register_member();
            af.Owner = this;
            af.Show();
        }

        private void pictureBox2_Click(object sender, EventArgs e)
        {
            Register_command af = new Register_command();
            af.Owner = this;
            af.Show();
        }

        private void завершитиЗмаганняToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            DialogResult dialogResult = MessageBox.Show("База даних буде очищена.
Продовжити?", "Увага!", MessageBoxButtons.YesNo);
            if (dialogResult == DialogResult.Yes)
            {
                string connectionString = "provider=Microsoft.Jet.OLEDB.4.0;Data
Source=dipolm.mdb";
                OleDbConnection dbConnection = new OleDbConnection(connectionString);
                dbConnection.Open();
```

```
                OleDbCommand dbCommand = new OleDbCommand("DELETE FROM comanda",
dbConnection);
                OleDbCommand dbCommand1 = new OleDbCommand("DELETE FROM EK600",
dbConnection);
                OleDbCommand dbCommand2 = new OleDbCommand("DELETE FROM EH600",
dbConnection);
                OleDbCommand dbCommand3 = new OleDbCommand("DELETE FROM EL600",
dbConnection);
                OleDbCommand dbCommand4 = new OleDbCommand("DELETE FROM F2U",
dbConnection);
                OleDbCommand dbCommand5 = new OleDbCommand("DELETE FROM F4A",
dbConnection);
                OleDbCommand dbCommand6 = new OleDbCommand("DELETE FROM EK600stand",
dbConnection);
                OleDbCommand dbCommand7 = new OleDbCommand("DELETE FROM EH600stand",
dbConnection);
                OleDbCommand dbCommand8 = new OleDbCommand("DELETE FROM EL600stand",
dbConnection);
                OleDbCommand dbCommand9 = new OleDbCommand("DELETE FROM F2Ustand",
dbConnection);
                OleDbCommand dbCommand10 = new OleDbCommand("DELETE FROM member",
dbConnection);
                dbCommand.ExecuteNonQuery();
                dbCommand1.ExecuteNonQuery();
                dbCommand2.ExecuteNonQuery();
                dbCommand3.ExecuteNonQuery();
                dbCommand4.ExecuteNonQuery();
                dbCommand5.ExecuteNonQuery();
                dbCommand6.ExecuteNonQuery();
                dbCommand7.ExecuteNonQuery();
                dbCommand8.ExecuteNonQuery();
                dbCommand9.ExecuteNonQuery();
                dbCommand10.ExecuteNonQuery();
                dbConnection.Close();
                MessageBox.Show("База даних була очищена!","Увага!");
            }
            else if (dialogResult == DialogResult.No)
            {
                //do something else
            }
        }

        private void довідкаToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            Process.Start("help.html");
        }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;

namespace AS_Zmagannya
{
    public partial class Register_command : Form
```

```
    {
        public Register_command()
        {
            InitializeComponent();
            //Заклинание для считывания и вывода даных с БД
            string connectionString = "provider=Microsoft.Jet.OLEDB.4.0;Data
Source=dipolm.mdb";
            OleDbConnection dbConnection = new OleDbConnection(connectionString);
            dbConnection.Open();
            string query = "SELECT * FROM comanda";
            OleDbCommand dbCommand = new OleDbCommand(query, dbConnection);
            OleDbDataReader dbReader = dbCommand.ExecuteReader();
            while (dbReader.Read())
            {
                dataGridView1.Rows.Add(dbReader["misto"], dbReader["CPO"],
dbReader["director"]);
            }
            dbReader.Close();
            dbConnection.Close();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string misto = misto_.Text;
            string CPO = CPO_.Text;
            string director = director_.Text;

            string connectionString = "provider=Microsoft.Jet.OLEDB.4.0;Data
Source=dipolm.mdb";
            OleDbConnection dbConnection = new OleDbConnection(connectionString);
            dbConnection.Open();
            //dataGridView1.Rows.Add();
            string query = "INSERT INTO comanda VALUES ('" + misto + "', '" + CPO +
"', '" + director + "')";
            OleDbCommand dbCommand = new OleDbCommand(query, dbConnection);

            if (dbCommand.ExecuteNonQuery() != 1)
                MessageBox.Show("Помилка виконання запиту!", "Помилка!");
            else
                MessageBox.Show("Дані додані!", "Увага!");
            dbConnection.Close();
            misto_.Text = "";
            CPO_.Text = "";
            director_.Text = "";
        }

        private void label5_MouseHover(object sender, EventArgs e)
        {
            label5.ForeColor = Color.Red;
        }

        private void label5_MouseLeave(object sender, EventArgs e)
        {
            label5.ForeColor = Color.Black;
        }

        private void label5_Click(object sender, EventArgs e)
        {
            if (dataGridView1.Visible == true)
            {
                dataGridView1.Visible = false;
            }
```

```
            else
                dataGridView1.Visible = true;
        }

        private void button2_Click(object sender, EventArgs e)
        {

        }
    }
}
```

ДОДАТОК 2

Презентація дипломної роботи

Київський Національний Університет Будівництва і Архітектури

Дипломна робота на тему «Автоматизована система"Судомодельні змагання"»

Виконав: Студент групи КНс-21

Чернишук О.М.

Керівник: Горда О.В.

Київ 2024

Актуальність розробки

- Простота реєстрації учасників
- Спрощення обрахунків балів
- Прозорість оцінювання
- Обрахунок стендового оцінювання;
- Обрахунок ходового оцінювання;

Мета розробки

Метою розробки даного проекту було створення десктопного застосунку для зруного та надійного процесу проведення змагань з судомодельного спорту.

Об'єктом дослідження є принцип проведення судомодельних змагань.

Предметом дослідження є технології застосунку і роботи з базами даних.

Використані технології під час розробки

Під час розробки для забезпечення найкращого результату були використані наступні технології:

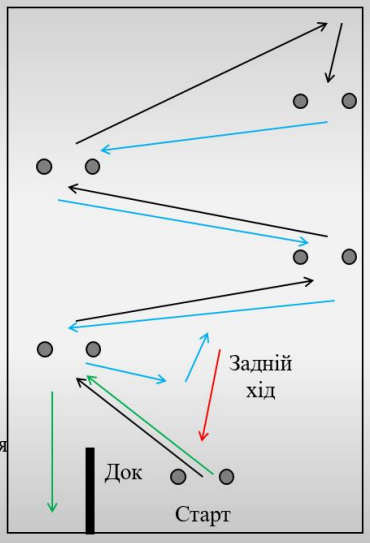
- .NET framework - платформа розробки програмного забезпечення
- WPS - технологія для розробки десктопних додатків з використанням графічного інтерфейсу користувача
- Access data base - база даних для зберігання інформації
- OleDB - бібліотека для роботи з базою даних

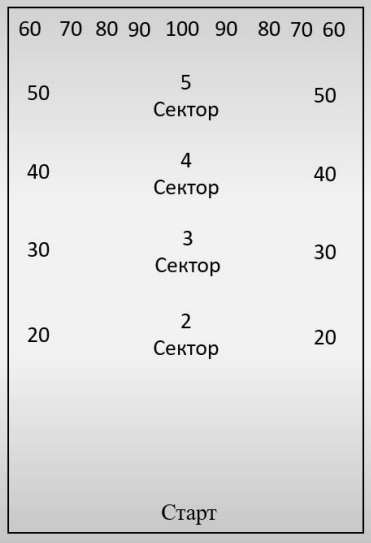
Процес проведення змагань

- Реєстрація учасників
- Стендове оцінювання (зовнішній вигляд моделі)
- Ходове оцінювання (ходові можливості)
- Підведення підсумків

Дистанції моделей

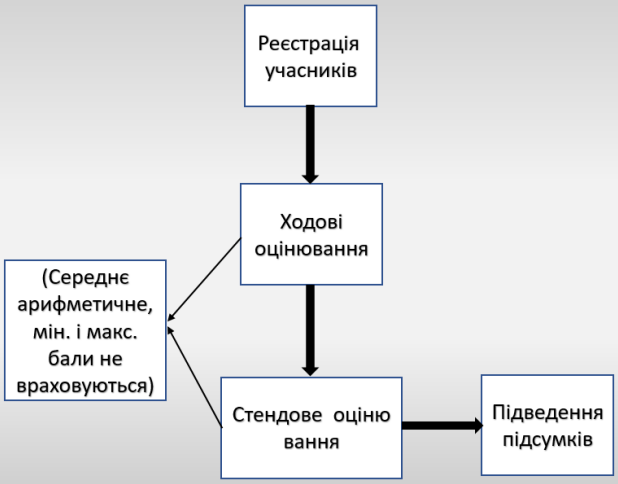
Класи
F4A
F2U



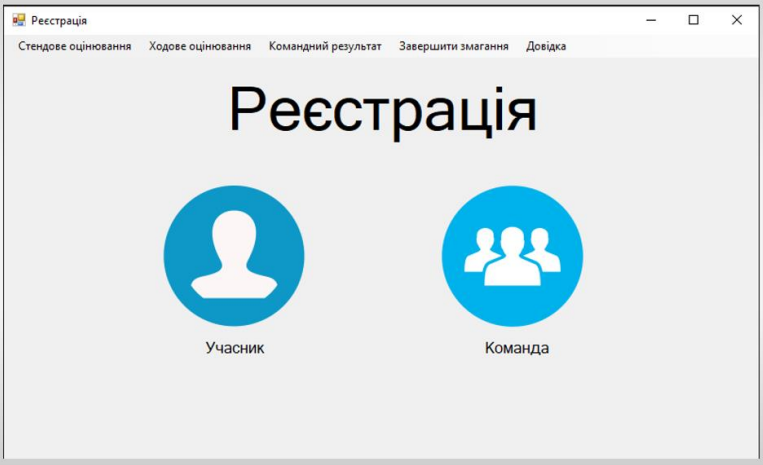


Класи
EK600
EH600
EL600

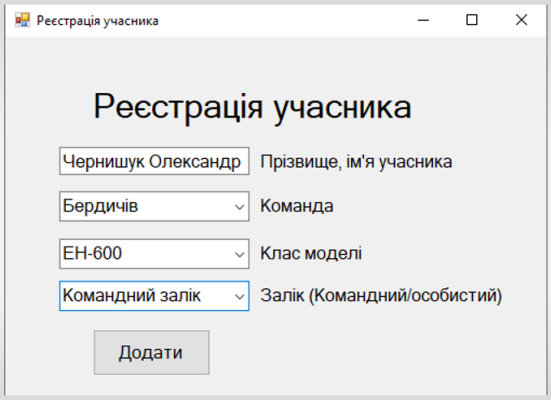
Алгоритм проведення змагань

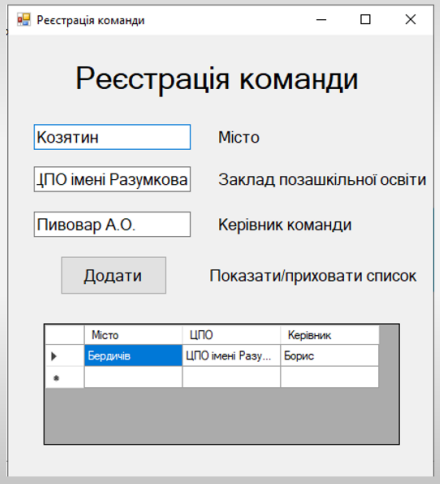


Приклад роботи застосунку (головна сторінка)



Приклад роботи застосунку (сторінки реєстрації учасників і команди)





Переваги розроблюваного веб-застосунку

- Автоматизація процесу: Десктопний застосунок дозволяє автоматизувати багато процесів, пов'язаних з моделюванням і симуляцією. Це включає в себе автоматичні обчислення, графічне відображення результатів, збереження даних і автоматичний аналіз.
- Точність і надійність: Порівняно з ручними розрахунками, де є високий ризик людської помилки, десктопний застосунок забезпечує більшу точність і надійність результатів. Алгоритми вбудовані в програмне забезпечення здатні працювати з високою точністю і повторюваністю.

- Швидкість обчислень: Десктопні програми зазвичай працюють швидше, ніж людина може робити розрахунки вручну. Це дозволяє ефективніше виконувати багато ітерацій та аналізувати більші обсяги даних за короткий час.

Висновки

У даному дипломному проекті була розроблена база даних та програмний застосунок автоматизованої стистеми визначення рейтингів учасників гуртка технічної творчості. Для побудови цієї системи були використані мова програмування C# та СУБД Access. У якості IDE була використана Microsoft Visual Studio.

Програма має наступний функціонал:

- Створення та редагування кошторисів;
- Створення звітів та їх друк;
- Управління користувачами та різнорівневий доступ;
- Додавання інформації у таблиці довідники;

Дякую за увагу!