Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Створення програмної системи для обліку даних шахових турнірів

Виконав: студент IV курсу, групи СН-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Місько А. П.
(підпис) (прізвище та ініціали)

Керівник _____ Фриз М. Є.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ _____
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _червня_ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Міський Андрій Петрович
(прізвище, ім'я, по батькові)

1. Тема роботи Створення програмної системи для обліку даних шахових турнірів

Керівник роботи Фриз Михайло Євгенович, кандидат технічних наук, доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*14*» *травня* 2026 року № *4/9-239*

2. Термін подання студентом завершеної роботи 22 червня 2026 р.

3. Вихідні дані до роботи Літературні та інтернет-джерела з технологій розробки програмних систем, C#, SQL, баз даних.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз предметної області та постановка завдання. 1.1 Особливості організації та автоматизації сучасних шахових змагань. 1.2 Огляд та порівняльний аналіз існуючих програмних рішень. 1.3 Обґрунтування вибору математичного апарату та технологій реалізації. 1.4 Висновок до першого розділу. 2. Проектування системи. 2.1 Вимоги до системи. 2.2 Діаграми прецедентів. 2.3 Проектування бази даних. 2.4 Моделювання динамічної поведінки та алгоритмічного забезпечення системи. 2.5 Висновок до другого розділу. 3. Реалізація та технології. 3.1 Розробка графічного інтерфейсу користувача. 3.2 Програмна реалізація бізнес-логіки та взаємодії з базою даних. 3.3 Тестування інформаційної системи. 3.4 Висновки до третього розділу. 4. Безпека життєдіяльності, основи охорони праці. 4.1 Психологічні причини нещасних випадків і травматизму. 4.2 Соціальне значення охорони праці. 4.3 Висновки до четвертого розділу. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титульна сторінка. 2. Мета кваліфікаційної роботи. 3. Актуальність обраної теми. 4. Аналіз предметної області. 5. Обґрунтування технологій. 6. Архітектура системи. 7. Проектування бази даних. 8. Алгоритмічне забезпечення системи. 9. Графічний інтерфейс. 10. Тестування та безпека системи. 11. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Гурик О. Я. кандидат технічних наук, доцент кафедри МТ		

7. Дата видачі завдання 27 січня 2026р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	27.01.2026	*Виконано*
2.	Підбір та опрацювання літературних джерел по темі кваліфікаційної роботи	27.01.2026-16.02.2026	*Виконано*
3.	Виконання дослідження щодо розробки програмного рішення для обліку шахових турнірів	18.02.2026-17.05.2026	*Виконано*
	Розроблення функціоналу програмної системи		
4.	Оформлення розділу «Аналіз предметної області та	19.05.2026-22.05.2026	*Виконано*
	постановка завдання»		
5.	Оформлення розділу «Проектування системи»	24.05.2026-28.05.2026	*Виконано*
6.	Оформлення розділу «Реалізація та технології»	28.05.2026-31.05.2026	*Виконано*
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	01.06.2026-06.06.2026	*Виконано*
8.	Виконання завдання до підрозділу «Основи охорони	01.06.2026-06.06.2026	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	06.06.2026-09.06.2026	*Виконано*
10.	Нормоконтроль		*Виконано*
11.	Перевірка на плагіат		*Виконано*
12.	Попередній захист кваліфікаційної роботи		*Виконано*
13.	Захист кваліфікаційної роботи	24.06.2026	

Студент _____ Місько А. П.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Фриз М. Є.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Створення програмної системи для обліку даних шахових турнірів // Кваліфікаційна робота освітнього рівня «Бакалавр» // Міську Андрій Петрович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-42 // Тернопіль, 2026 // С. 60, рис. – 14, табл. – 1, додат. – 2, бібліогр. – 35.

Ключові слова: шаховий турнір, інформаційна система, автоматизація обліку, база даних, жеребкування, система рейтингу, проектування програмного забезпечення, алгоритми розподілу пар.

Кваліфікаційна робота присвячена дослідженню та розробці програмної системи для автоматизації обліку даних шахових турнірів.

В першому розділі досліджено специфіку проведення шахових змагань та проаналізовано існуючі аналоги. Розглянуто основні турнірні системи та математичні моделі розрахунку рейтингу гравців. Виявлено недоліки ручного керування даними та обґрунтовано доцільність розробки спеціалізованого програмного забезпечення.

В другому розділі спроектовано архітектуру системи та визначено функціональні вимоги. Розроблено UML-діаграми прецедентів і станів, а також схему бази даних, що забезпечує цілісність інформації про гравців, тури та результати партій.

В третьому розділі описано процес програмної реалізації системи та вибір технологічного стеку. Проаналізовано роботу алгоритмів жеребкування та розрахунку турнірних показників. Проведено тестування модулів на коректність обробки даних та стійкість до помилок користувача.

ANNOTATION

Development of a software system for chess tournament data accounting // Qualification work of the educational level «Bachelor» // Misko Andrii Petrovych // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-42 // Ternopil, 2026 // P.60, fig. – 14, tabl. – 1, annexes. – 2, references – 35.

Keywords: chess tournament, information system, accounting automation, database, matchmaking, rating system, software engineering, pairing algorithms.

The qualification work is dedicated to the research and development of a software system for automating the accounting of chess tournament data.

The goal of the work is to create a functional software solution that simplifies the organization of chess competitions, ensures accurate data storage, and automates the process of pairing and result calculation.

The first section investigates the specifics of chess competitions and analyzes existing digital analogues. The main tournament systems and mathematical models for calculating player ratings are reviewed. The disadvantages of manual data management are identified, and the expediency of developing specialized software is justified.

The second section focuses on the system architecture design and functional requirements. UML use case and state diagrams were developed, alongside a database schema that ensures the integrity of data regarding players, rounds, and match results.

The third section describes the software implementation and the selection of the technology stack. The performance of pairing algorithms and tournament metrics calculation is analyzed. The system modules were tested for data processing accuracy and user error tolerance.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ADO.NET (ActiveX Data Objects .NET) – технологія доступу до даних, що є частиною платформи .NET Framework.

ER (Entity-Relationship) – модель «сутність-зв’язок», яка використовується для проектування реляційних баз даних.

GUI (Graphical User Interface) – графічний інтерфейс користувача.

IDE (Integrated Development Environment) – інтегроване середовище розробки програмного забезпечення.

PGN (Portable Game Notation) – стандартний комп'ютерний формат для запису шахових партій.

SQL (Structured Query Language) – структурована мова запитів для створення, модифікації та управління даними в реляційній базі даних.

UML (Unified Modeling Language) – уніфікована мова графічного моделювання для розробки архітектури програмного забезпечення.

БД – база даних.

БЖД – безпека життєдіяльності.

ЕОМ – електронно-обчислювальна машина.

ЗМІСТ

ВСТУП

Актуальність теми. Внаслідок глобалізації та стрімкої цифровізації спортивної індустрії, традиційні методи організації змагань поступаються місцем інтегрованим інформаційним системам. Шахи, як вид спорту з чіткими математичними алгоритмами жеребкування та рейтингування, потребують високої точності обробки даних. Проте існуючі рішення часто є складними у використанні для невеликих клубів або мають закритий код. Тому розробка спеціалізованої програмної системи для обліку даних шахових турнірів є актуальним напрямком сучасних досліджень в галузі автоматизації спортивного менеджменту та проектування інформаційних систем.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є підвищення якості адміністрування шахових змагань шляхом автоматизації процесів реєстрації учасників, формування турнірних пар та розрахунку результатів. Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

- проаналізувати стан досліджень у сфері автоматизації шахових турнірів та існуючі програмні аналоги;
- дослідити математичні моделі жеребкування та алгоритми розрахунку індивідуальних рейтингів (система Ело);
- спроектувати архітектуру програмної системи та структуру бази даних для надійного збереження турнірної інформації;
- розробити програмне забезпечення з інтуїтивно зрозумілим інтерфейсом користувача;
- провести тестування системи на реальних сценаріях проведення турнірів для перевірки коректності роботи алгоритмів.

Практичне значення одержаних результатів полягає у створенні функціонального програмного продукту, який дозволяє автоматизувати роботу арбітрів та організаторів турнірів, Розроблену систему можна впроваджувати у діяльність шахових шкіл, клубів та громадських організацій.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ.

1.1 Особливості організації та автоматизації сучасних шахових змагань

Організація сучасних шахових змагань – це складний багаторівневий процес, що поєднує в собі елементи спортивного адміністрування, суворе дотримання міжнародних регламентів (зокрема правил ФІДЕ) [1] та інтенсивну обробку динамічних інформаційних масивів. На відміну від багатьох інших видів спорту, де фіксація результатів є переважно лінійною, шахи критично залежать від точності алгоритмічного супроводу на кожному етапі проведення турніру. Це зумовлено математичним характером побудови змагального процесу, де кожен крок організатора чи арбітра регламентується суворими логічними правилами.

Ключові особливості організації змагань:

1. Інформаційна насиченість. Кожен гравець має унікальний набір кваліфікаційних атрибутів (прізвище, ім'я, по батькові, спортивний розряд або звання, клубна приналежність, поточні національний та міжнародний рейтинги), які повинні бути обов'язково верифіковані перед початком змагання. Велика кількість первинних даних вимагає наявності надійних механізмів валідації та збереження інформації для запобігання дублюванню або спотворенню профілів учасників [2].

2. Складність жеребкування. Процес формування ігрових пар (pairing) у кожному наступному турі безпосередньо залежить від результатів попередніх ігор, поточного набраного кожним учасником очкового доробку, кольору фігур (білі або чорні), якими грав учасник у попередніх раундах, а також наявності штрафних очок або технічних пропусків [3]. Необхідність дотримання балансу та чергування кольору фігур, а також недопущення повторних зустрічей між тими самими шахістами в межах одного турніру робить алгоритми жеребкування багатокритеріальними. Виконання цих правил вручну в турнірах,

де кількість учасників перевищує 20 осіб, призводить до колосальних часових витрат, затягування регламенту змагань та високої імовірності виникнення критичних технічних помилок з боку суддівської колегії [4].

3. Динамічність рейтингів. Результат кожної зіграної партії безпосередньо впливає на поточний статус гравця в загальній ієрархії, змінюючи його кваліфікаційну силу. Автоматизація дозволяє миттєво розраховувати очікуваний результат партії на основі різниці стартових показників опонентів та фіксувати безпосередню зміну індивідуального рейтингу за математичною системою Ело [5] відразу після внесення результату гри в базу даних. Оскільки формування ігрових пар та динаміка зміни рейтингів базуються на імовірнісних моделях, доцільно використовувати сучасні методи математичної статистики та аналізу випадкових процесів для коректної обробки таких даних [6, 7].

Процес автоматизації шахового менеджменту зазвичай охоплює три основні взаємопов'язані стадії:

1. Дотурнірна стадія. Створення електронної бази учасників поточного змагання, реєстрація нових шахістів, перевірка їхньої легітурності та відповідності заявленому регламенту, а також автоматизований розподіл гравців за стартовими номерами згідно з їхнім поточним рейтинговим показником (первинне сортування або посів).

2. Турнірна стадія. Циклічна генерація турнірних таблиць та протоколів для кожного раунду, оперативна фіксація результатів завершених партій, моніторинг поточної ситуації та автоматичний перерахунок складних додаткових коефіцієнтів (таких як коефіцієнт Бухгольца, Зоннеборна-Бергера або прогресивний показник), що є вирішальними при розподілі фінальних місць у разі абсолютної рівності набраних учасниками очок.

3. Посттурнірна стадія. Автоматичне формування підсумкових звітних протоколів, відомостей для присвоєння чергових спортивних розрядів, вивантаження структурованих результатів для обрахунку офіційних рейтингів у вищі кваліфікаційні комісії та експорт різноманітних статистичних даних для аналізу результативності гравців [8].

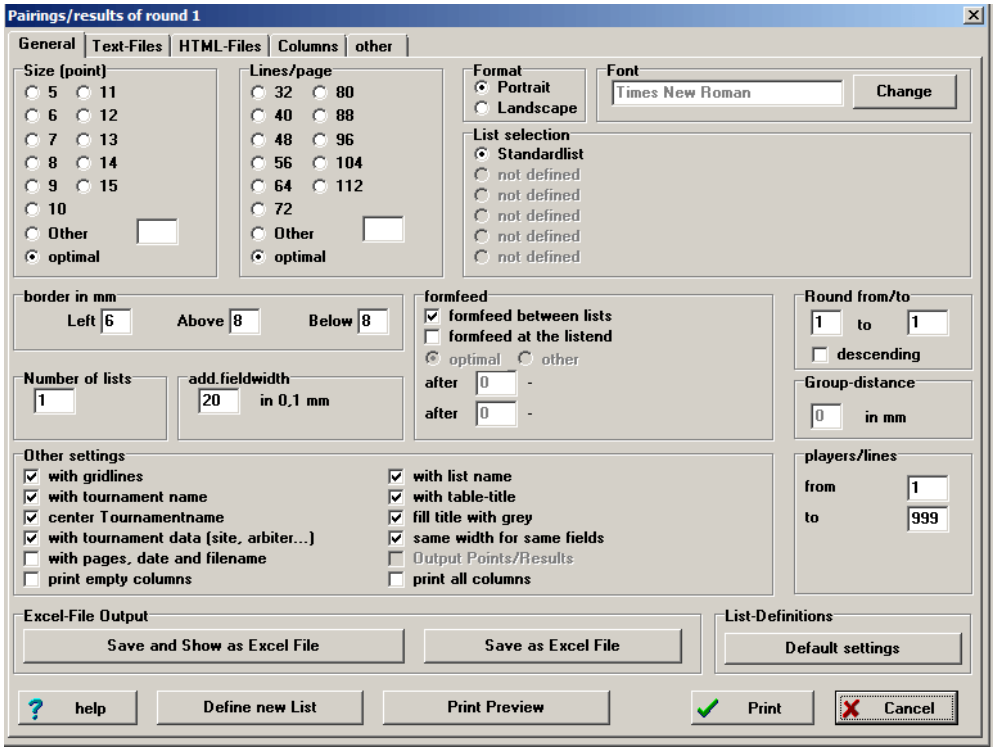
Використання спеціалізованих програмних систем дозволяє повністю мінімізувати вплив суб'єктивного людського фактора, забезпечити абсолютну прозорість і неупередженість суддівства та значно прискорити проведення змагань на всіх етапах [9]. Таким чином, створення гнуцької, автономної та доступної системи для обліку даних є необхідним і актуальним кроком для комплексної цифровізації шахової діяльності на локальному та регіональному рівнях [10].

1.2 Огляд та порівняльний аналіз існуючих програмних рішень

На сьогоднішній день на ринку спеціалізованого програмного забезпечення існує кілька визнаних систем для автоматизації шахових змагань, які активно використовуються арбітрами та організаторами. Проте кожне з наявних рішень має низку суттєвих обмежень (функціональних, інтерфейсних або фінансових), що безпосередньо обґрунтовує доцільність розробки власного програмного комплексу, оскільки вибір архітектури ПЗ є критичним етапом життєвого циклу розробки будь-якої складної інформаційної системи.

Найбільш поширеним професійним десктопним продуктом, офіційно схваленим Міжнародною шаховою федерацією, є програма Swiss-Manager. Вона підтримує практично всі відомі системи жеребкування (колову, швейцарську, систему на виліт тощо) та володіє розгалуженим набором конфігурацій для проведення турнірів будь-якого масштабу. До беззаперечних переваг Swiss-Manager належать висока надійність алгоритмів обробки великих масивів даних та повна інтеграція з міжнародним сервером Chess-Results для публікації поточних звітів. Проте критичними недоліками цієї системи є морально застарілий графічний інтерфейс користувача, який зображений на рисунку 1.1, що суперечить сучасним принципам людино-машинної взаємодії [11]. Зовнішній вигляд програми не змінювався протягом багатьох років, надзвичайно високий поріг входження для новачків, складність у навчанні персоналу, закритий вихідний код, а також висока вартість комерційної

ліцензії, що робить її малодоступною для невеликих локальних клубів чи навчальних закладів.



Рисуно 1.1 – Інтерфейс програми Swiss-Manager

Альтернативним сучасним напрямком є веб-орієнтовані хмарні платформи, серед яких виділяється система Tornelo, логотип якої зображений на рисунку 1.2. Ця платформа орієнтована на проведення онлайн та гібридних шахових змагань, володіє сучасним ергономічним дизайном інтерфейсу та містить інтегровані алгоритми прокторингу для захисту від нечесної гри (чітерства) [12]. Разом з тим, функціонування Tornelo критично залежить від стабільного інтернет-з’єднання, а її фінансова модель передбачає постійну регулярну оплату за кожного зареєстрованого в турнірі учасника, що є економічно обтяжливим для некомерційних або бюджетних організацій.

Рисунок 1.2 – Логотип платформи Tornelo

Також у європейському просторі відома система ChessArbiter, яка демонструє суворе відповідність шаховому кодексу, проте характеризується обмеженими можливостями кастомізації інтерфейсу під конкретні потреби користувача та закритою структурою локальної бази даних, що ускладнює інтеграцію з іншими інформаційними системами [13]. На рисунку 1.3 зображено головний вебсайт платформи.

Рисунок 1.3 – Вебсайт платформи ChessArbiter

Для наочного порівняння існуючих рішень та обґрунтування розробки власної системи складено порівняльну таблицю 1.1, яка структурує особливості всіх ПЗ згідно з методологією порівняльного аналізу інженерних рішень.

Таблиця 1.1 - Порівняльна характеристика програмних засобів

Критерій порівняння	**Swiss-Manager**	**Tornelo**	**ChessArbiter**
Тип архітектури	Desktop	Cloud/SaaS	Desktop
Зручність інтерфейсу	Низька	Висока	Середня
Доступність (ціна)	Платна	Передплата	Платна
Офлайн режим	Повний	Обмежений	Повний
Кросплатформеність	Лише Windows	Будь-яка (браузер)	Лише Windows

Проведений критичний аналіз існуючих рішень демонструє чітко виражений розрив між складними професійними системами та сучасними веб-платформами. Ринок програмного забезпечення потребує універсального продукту, який би поєднував у собі високу ергономіку та інтуїтивно зрозуміле керування з архітектурною стійкістю та можливістю повноцінної автономної роботи в умовах відсутності мережевого з’єднання [14]. Актуальним є створення системи, яка забезпечить не лише автоматизацію рутинних процесів жеребкування, а й гарантуватиме цілісність даних та доступність функціоналу для невеликих шахових спільнот, клубів та освітніх закладів. Такий підхід дозволить демократизувати інструментарій для організації змагань, мінімізувати експлуатаційні витрати та забезпечити високий рівень супроводу турнірів без необхідності тривалого спеціалізованого навчання персоналу.

1.3 Обґрунтування вибору математичного апарату та технологій реалізації

Ефективність та об'єктивність функціонування розроблюваної системи обліку даних шахових турнірів безпосередньо залежить від обраних алгоритмів обробки результатів, логічних моделей та архітектурних рішень. Основним інструментом для оцінювання поточної сили гравців та статистичного аналізу результатів у часі було обрано міжнародну рейтингову систему Ело. На відміну від найпростіших систем лінійного накопичення очок, де статус учасника зростає однаково незалежно від рівня суперника, цей метод дозволяє динамічно та гнучко коригувати кваліфікаційний показник шахіста на основі відносної складності кожного зіграного матчу.

Логіка системи Ело базується на порівнянні поточної сили опонентів перед початком партії. Програма автоматично визначає ймовірнісний очікуваний результат зустрічі: гравець із вищим рейтингом має вищі математичні шанси на перемогу. Після завершення партії система порівнює фактичний підсумок з очікуваним, використовуючи методи теорії ймовірностей, що дозволяє уникнути викривлень при оцінюванні. Оскільки процеси зміни рейтингів мають імовірнісну природу, ми спираємося на сучасні підходи до аналізу властивостей випадкових процесів, зокрема описані у дослідженнях [15]. Інтеграція такої моделі у програмний комплекс забезпечує високу об'єктивність оцінювання результативності гравців.

З точки зору створення програмних систем, для управління великими масивами взаємопов’язаних даних (відомості про гравців, протоколи турів, результати партій) найдоцільнішим є використання реляційної моделі баз даних [16]. Це гарантує суворе дотримання транзакційних стандартів, цілісність інформації при потенційних збоях та відсутність надмірності даних. Використання системи управління базами даних Microsoft SQL Server дозволяє проектувати оптимізовані запити мовою T-SQL для миттєвого розрахунку поточних лідербордів [17], автоматичного визначення набраних балів та

швидкого формування аналітичної звітності безпосередньо під час перебігу змагання.

Для реалізації клієнтської частини системи обрано об’єктно-орієнтований підхід та мову програмування C# (платформа .NET) [18], що дозволяє чітко структурувати логіку програми за принципом модульності. Виділення сутностей предметної області (гравець, турнір, матч) в окремі класи забезпечує високу читабельність коду, можливість застосування патернів проектування [19] та полегшує подальше масштабування системи. Поєднання перевірених часом моделей та сучасних стандартів розробки програмного забезпечення, зокрема підходів до об’єктно-орієнтованої ідентифікації систем [20], створює надійне й архітектурно стійке підґрунтя для розв’язання поставленої задачі.

1.4 Висновок до першого розділу

У першому розділі кваліфікаційної роботи було проведено комплексний аналіз предметної області, що стосується особливостей організації, адміністрування та автоматизації сучасних шахових змагань. На основі виконаного дослідження зроблено такі висновки:

1. Обґрунтовано необхідність автоматизації. Виявлено, що сучасні турніри характеризуються великими обсягами даних та високою складністю алгоритмів жеребкування, що створює ризик технічних помилок і підтверджує актуальність розробки спеціалізованого ПЗ.

2. Проаналізовано існуючі аналоги. Огляд систем (Swiss-Manager, Tornelo, ChessArbiter) показав наявність ніші для локального, автономного та доступного продукту, який би відповідав вимогам сучасного UI/UX.

3. Визначено математичний апарат. Для забезпечення об'єктивності обрано систему Ело, що базується на статистичних та імовірнісних підходах до обробки даних [21], дозволяючи виконувати динамічний перерахунок коефіцієнтів.

4. Сформульовано вимоги до розробки. Встановлено, що система повинна базуватися на реляційній моделі даних (SQL Server) та гнучкій архітектурі на базі C# [22], що забезпечить надійну роботу з даними в умовах турніру.

Таким чином, результати детального аналізу предметної області є вичерпними та дозволяють повноцінно перейти до наступного етапу кваліфікаційної роботи – системного проектування загальної архітектури системи, розробки реляційної структури бази даних, створення UML-моделей та детального опису функціональних можливостей програмного продукту.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Вимоги до системи

На етапі проектування інформаційної системи для обліку даних шахових турнірів критично важливим є чітке формулювання та структурування вимог до програмного забезпечення. Ці вимоги виступають основою для формування архітектури додатку і розподіляються на дві основні категорії: функціональні, що визначають безпосередні можливості програми та перелік операцій користувача, та нефункціональні, які описують якісні характеристики, обмеження та експлуатаційні параметри системи.

Функціональні вимоги описують конкретні дії та сценарії, які користувач (адміністратор системи або спортивний суддя) зможе виконувати в межах програмного комплексу:

1. Управління реєстром гравців. Забезпечення можливості додавання нових учасників до загальної бази даних, редагування існуючих персональних та кваліфікаційних профілів, а також видалення застарілих або помилково внесених відомостей із контролем цілісності зв'язків.

2. Конфігурація турнірів. Створення нових змагальних подій із можливістю гнучкого задання їхніх атрибутів, таких як унікальна назва, дата і місце проведення, а також безпосередній вибір базового алгоритму жеребкування (колова система або система на виліт/плей-оф) залежно від затвердженого регламенту.

3. Генерація турів. Автоматичне формування ігрових пар для кожного поточного раунду на основі накопичених результатів попередніх ігор, а також суворе математичне дотримання правил чергування та балансу кольорів фігур для кожного окремого шахіста.

4. Фіксація результатів матчів. Забезпечення зручного інтерфейсу для оперативного введення підсумків завершених партій (перемога, нічия, поразка) та миттєве автоматичне оновлення поточної турнірної таблиці.

5. Розрахунок додаткових показників. Автоматичний перерахунок поточного турнірного становища (лідерборду) із можливістю відображення динамічно нарахованих очок та коригування відносного статусу учасників безпосередньо під час перебігу змагання.

Нефункціональні вимоги визначають якісні характеристики, технічні критерії та умови стабільності функціонування розроблюваного програмного продукту:

1. Надійність. Забезпечення абсолютного збереження та консистентності даних у реляційній базі SQL Server при раптовому чи аварійному завершенні роботи клієнтського додатка, збоях операційної системи або критичних проблемах із живленням комп'ютерного обладнання [23].

2. Зручність використання (Usability). Проектування ергономічного графічного інтерфейсу на базі технології Windows Forms, що гарантує швидкий інтуїтивний доступ до ключових функцій керування турніром, мінімальну кількість переходів між вікнами та чітке візуальне розділення інформаційних зон.

3. Сумісність та системне середовище. Забезпечення стабільного функціонування та повної сумісності програмного комплексу в операційних системах сімейства Windows (зокрема версій Windows 10 та Windows 11) без необхідності встановлення додаткового високовартісного проміжного софту.

4. Ефективність та швидкодія. Висока швидкість обробки даних та виконання алгоритмів вибірки гравців, формування транзакційних запитів до бази даних і генерації звітів з мінімальним часом відгуку інтерфейсу навіть при максимальному навантаженні на систему.

2.2 Діаграми прецедентів

Для візуалізації функціональної структури, архітектури та внутрішньої логіки розроблюваної інформаційної системи використано мову графічного моделювання UML. Побудова концептуальних моделей дозволяє перейти від

словесного опису вимог до чітного розуміння того, як взаємодіють між собою програмні компоненти та кінцеві користувачі.

Діаграма прецедентів (Use Case Diagram) відображає взаємодію основного актора – судді турніру або адміністратора – із програмним комплексом. Основними прецедентами, що визначають бізнес-логіку змагань, є управління глобальним реєстром гравців, створення та конфігурація нового турніру із заданням системи жеребкування, а також реєстрація обраних учасників на поточне змагання. Діаграма зображена на рисунку 2.1. На етапі проведення турніру ключовими прецедентами виступають автоматична генерація ігрових пар для поточного туру, внесення результатів завершених партій (перемога, нічия, поразка) та динамічне формування підсумкового звіту, що реалізується через миттєвий розрахунок турнірної таблиці (лідерборду). Кожен прецедент описує окремий, логічно завершений функціональний блок, який забезпечує автоматизацію рутинних операцій арбітра.

Рисунок 2.1 – Діаграма прецедентів

Діаграма класів (Class Diagram), яка зображена на рисунку 2.2, та структурна модель програми визначають внутрішню архітектуру додатка

мовою об'єктно-орієнтованого програмування C# [24]. Оскільки клієнтська частина системи побудована на базі технології Windows Forms із використанням ADO.NET, архітектура поєднує класи візуальних форм із сутностями предметної області [25] та включає такі основні класи:

1. Player (Гравець) – базова сутність предметної області, яка містить ключові атрибути учасника змагань. Клас включає поля для збереження унікального ідентифікатора (PlayerID), повного імені шахіста (FullName), його дати народження (BirthDate), поточного рейтингу Ело (CurrentRating), кваліфікаційного розряду (Rank) та назви шахового клубу (Club), який він представляє.

2. Tournament (Турнір) – сутність, що відповідає за конфігурацію та стан спортивного заходу. Вона містить унікальний номер (TournamentID), назву змагання (TournamentName), тип системи жеребкування (SystemType), часові рамки проведення (StartDate, EndDate), обраний контроль часу (TimeControl), а також логічний прапорець поточного статусу турніру (IsActive). Клас слугує базисом для реєстрації гравців і виклику алгоритмів формування ігрових раундів.

3. Match (Матч) – клас, що описує безпосередню ігрову взаємодію між двома учасниками в конкретному турі. Він оперує ідентифікатором самого матчу (MatchID), зв'язком із турніром (TournamentID), номером поточного раунду (RoundNumber), унікальними номерами гравців білими та чорними фігурами (WhitePlayerID, BlackPlayerID), зафіксованим результатом партії (Result), прапорцем завершення гри (IsPlayed) та датою проведення матчу (MatchDate).

4. Класи взаємодії з БД – виконання транзакційних SQL-запитів та отримання даних із бази SQL Server використовуються оптимізовані системні класи SqlConnection та SqlCommand, що забезпечують швидкий прямий обмін інформацією без надлишкових архітектурних прошарків [26].

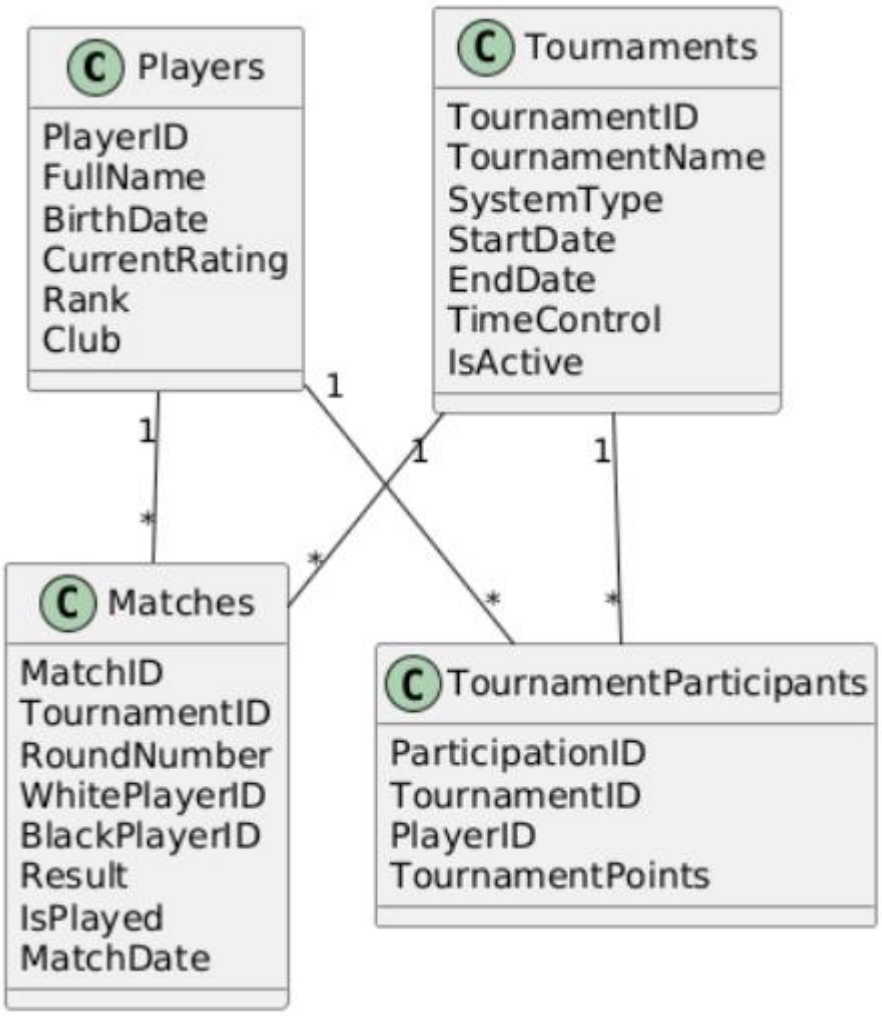


Рисунок 2.2 – Діаграма класів програмної системи

Використання описаних діаграм та структурних моделей, зокрема принципів об’єктно-орієнтованої ідентифікації структур, дозволяє здійснити плавний перехід від абстрактного опису предметної області до безпосередньої програмної реалізації, забезпечуючи чітку, зрозумілу та легко масштабовану архітектуру майбутнього настільного додатку.

2.3 Проектування бази даних

Для надійного збереження, ефективного пошуку та забезпечення цілісності інформації в інформаційній системі розроблено структуру реляційної бази даних. Процес проектування включає формування концептуальної схеми у вигляді ER-діаграми (Entity-Relationship), яка визначає ключові сутності

предметної області, їхні атрибути та логічні взаємозв’язки між ними. Побудована архітектура даних орієнтована на ефективне виконання транзакційних запитів та унеможливлення аномалій модифікації чи видалення інформації.

Основними сутностями розробленої системи є:

1. Гравці (Players) – сутність, призначена для збереження персональних, реєстраційних та кваліфікаційних відомостей про шахістів. Таблиця містить такі атрибути:

- PlayerID (INT, Primary Key) – унікальний системний ідентифікатор гравця, що генерується автоматично;
- FullName (NVARCHAR) – прізвище, ім'я та по батькові учасника змагань;
- Rank (NVARCHAR) – поточний спортивний розряд або звання шахіста (наприклад, КМС, І розряд);
- CurrentRating (INT) – актуальний кваліфікаційний рейтинг Ело, що динамічно змінюється за результатами зіграних партій.

2. Турніри (Tournaments) – сутність, яка описує організаційні та регламентні параметри конкретних шахових змагань. До її складу входять такі поля:

- TournamentID (INT, Primary Key) – унікальний цифровий ідентифікатор турніру;
- TournamentName (NVARCHAR) – офіційна назва шахового змагання;
- SystemType (NVARCHAR) – текстовий опис системи проведення турніру, що визначає логіку жеребкування раундів (зокрема, «Колова» або «На виліт / Плей-оф»);
- StartDate (DATETIME) – календарна дата та час офіційного старту заходу.

3. Учасники турнірів (TournamentParticipants) – проміжна сутність, що реалізує зв'язок типу «багато-до-багатьох» між гравцями та турнірами [27], фіксуючи поточний склад учасників для кожного конкретного змагання:

- TournamentID (INT, Foreign Key) – ідентифікатор змагання, пов'язаний із таблицею турнірів;
- PlayerID (INT, Foreign Key) – ідентифікатор шахіста, пов'язаний із глобальною таблицею гравців.

4. Партії/Матчі (Matches) – центральна транзакційна сутність, яка фіксує результати безпосередніх зустрічей шахістів у межах змагання та забезпечує зв'язок між гравцями, раундами та турнірами. Вона включає такі атрибути:

- MatchID (INT, Primary Key) – унікальний ідентифікатор конкретної зіграної чи згенерованої партії;
- TournamentID (INT, Foreign Key) – зовнішній ключ для логічного зв'язку матчу з відповідним турніром;
- WhitePlayerID (INT, Foreign Key) – ідентифікатор гравця, який за жеребкуванням керує білими фігурами;
- BlackPlayerID (INT, Foreign Key) – ідентифікатор гравця, який за жеребкуванням керує чорними фігурами;
- RoundNumber (INT) – порядковий номер поточного туру змагання;
- Result (FLOAT, Nullable) – підсумковий результат партії (приймає значення 1.0 при перемозі білих, 0.0 при перемозі чорних, 0.5 у разі нічиєї та залишається порожнім до моменту завершення гри);
- IsPlayed (BIT) – логічний прапорець, який вказує, чи було завершено матч та внесено його результат;
- MatchDate (DATETIME) – системна дата та час генерації або проведення поєдинку.

Логіка взаємозв'язків (Relationships) між таблицями базується на суворому забезпеченні референційної цілісності даних. Принцип роботи логіки БД зображено на рисунку 2.3. В основі архітектури схеми лежать зв'язки типу «один-до-багатьох» (1:N). Один турнір може містити велику кількість згенерованих матчів, так само як і кожен окремий гравець потенційно виступає учасником багатьох поєдинків протягом своєї діяльності. Завдяки

впровадженню сполучної таблиці TournamentParticipants забезпечується коректне розв'язання зв'язку «багато-до-багатьох» між сутностями шахістів та змагань, що дозволяє динамічно фільтрувати доступних у базі гравців та формувати поточний склад учасників у клієнтському додатку [28].

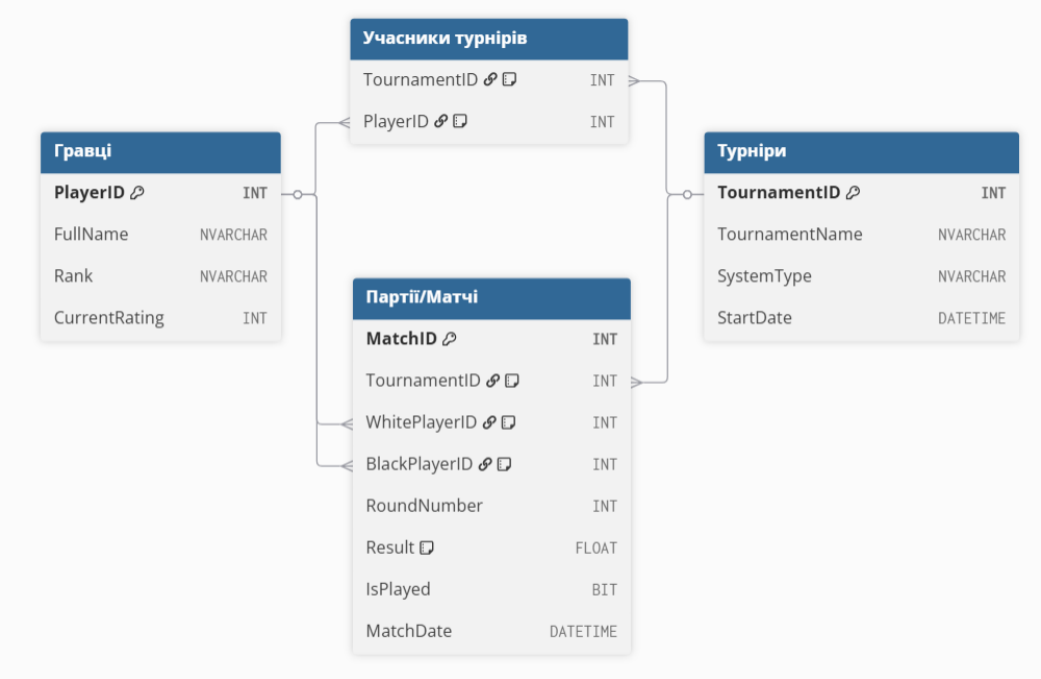


Рисунок 2.3 – ER-діаграма

Проектування схеми за правилами нормалізації (до третьої нормальної форми включно) дозволяє повністю мінімізувати дублювання та надмірність інформації, гарантує консистентність бази даних при транзакційних операціях та забезпечує максимально високу швидкодію при виконанні агрегуючих SQL-запитів для побудови динамічних лідербордів та підрахунку балів, що відповідає сучасним вимогам до проектування високонавантажених систем

2.4 Моделювання динамічної поведінки та алгоритмічного забезпечення системи

Для переходу від статичного опису структури даних та класів до безпосереднього кодування необхідно детально спроектувати динамічну

поведінку компонентів інформаційної системи під час виконання ключових операцій користувача. Основними процесами, що визначають динаміку системи обліку шахових турнірів, є процеси взаємодії між користувацьким інтерфейсом (клієнтською частиною додатку) та реляційною базою даних під час генерації нових раундів та фіксації результатів матчів.

Для моделювання часової послідовності обміну повідомленнями між об'єктами системи було розроблено діаграму послідовності UML (Sequence Diagram). Ця модель, яка зображена на рисунку 2.4, дозволяє простежити покроковий життєвий цикл виконання операцій та оптимізувати архітектуру взаємодії компонентів.

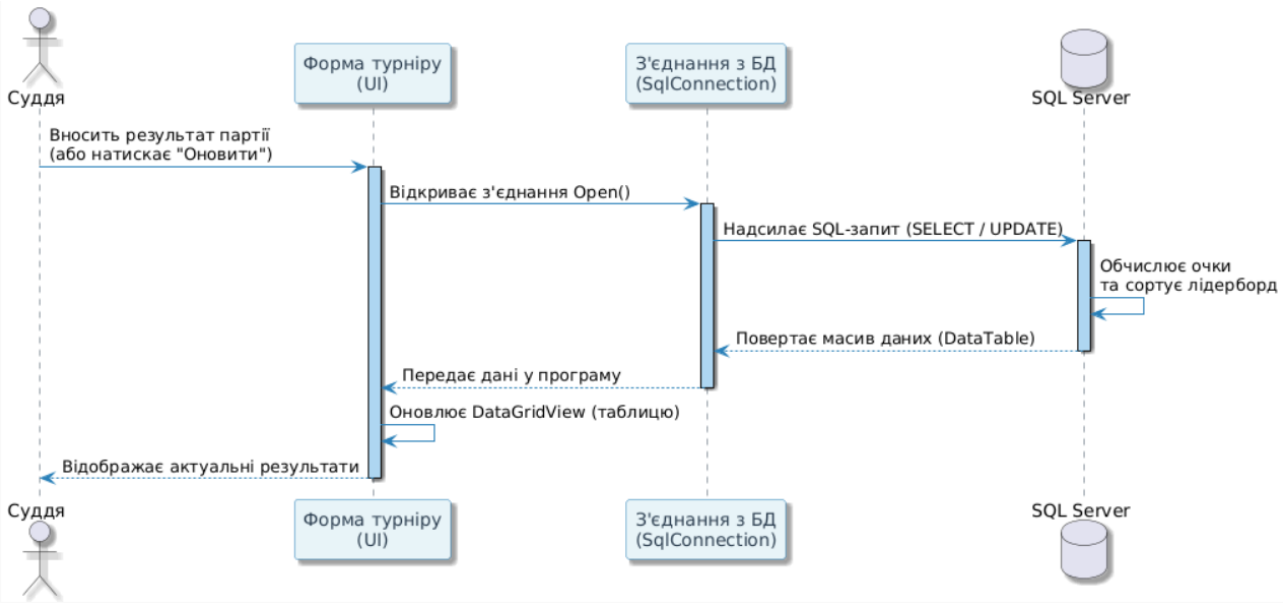


Рисунок 2.4 – Діаграма послідовності UML

Основними компонентами, що беруть участь у динамічній взаємодії під час роботи з турнірними результатами, є:

- оператор (спортивний арбітр або адміністратор системи), який ініціює дії через графічний інтерфейс;
- об'єкт візуального інтерфейсу (форма керування змаганням TournamentManagementForm), що фіксує події введення даних та відображає інформацію користувачу;

- об'єкт діалогового вікна матчів (MatchesListForm), який відповідає за відображення списку поточних ігор та внесення результатів партій;
- системні компоненти доступу до даних (SqlConnection та SqlCommand), які забезпечують транспортний рівень та безпечне виконання запитів;
- сервер бази даних (Microsoft SQL Server), що здійснює безпосередню обробку транзакцій, збереження інформації та агрегацію результатів.

Динаміка взаємодії під час оновлення турнірних балів та лідерборду відбувається за чітким каскадним сценарієм. Користувач у вікні списку ігор вносить результат шахової партії та закриває діалогову форму. У цей момент об'єкт головної форми перехоплює подію завершення роботи дочірнього вікна та автоматично ініціює виклик внутрішнього методу завантаження рейтингової таблиці. Даний метод відкриває асинхронне або синхронне з'єднання з базою даних через екземпляр класу підключення, формує параметризований текстовий запит до сервера та передає унікальний ідентифікатор поточного турніру. Сервер баз даних приймає запит, виконує обчислення накопичених очок за умовними операторами вибору, сортує гравців за спаданням балів та повертає структурований масив даних (таблицю результатів) назад у клієнтський додаток. Після отримання масиву даних об'єкт форми прив'язує його до елемента графічної сітки, оновлюючи відображення місць учасників на екрані, а з'єднання з базою даних закривається для звільнення системних ресурсів.

Важливою складовою проектування є опис алгоритмічного забезпечення системи, що відповідає за автоматичне жеребкування та генерацію нових турів. Функціонування алгоритму генерації пар базується на послідовній перевірці станів системи і представлене у вигляді логічної блок-схеми на рисунку 2.5, яка визначає порядок виконання операцій.

Рисунок 2.5 – Блок-схема алгоритму жеребкування

Процес формування нового туру розпочинається за запитом користувача при натисканні відповідного керуючого елемента. Першим етапом алгоритму є виконання валідаційного запиту до бази даних для перевірки наявності

незіграних матчів у попередньому раунді. Система аналізує поле результатів у таблиці матчів для поточного ідентифікатора турніру та максимального номера туру. Якщо алгоритм виявляє хоча б один запис, де результат партії не зафіксовано (значення рівне відсутності даних), процес генерації негайно переривається. Клієнтська форма блокує подальші дії та виводить на екран попереджувальне повідомлення про необхідність внесення всіх результатів поточного раунду. Це унеможливлює порушення хронології змагань та забігання наперед, що є суворим правилом шахового кодексу. Це унеможливлює порушення хронології змагань, що є критично важливим для цілісності спортивного процесу.

Якщо перевірка підтверджує, що всі попередні матчі успішно завершено та внесено до бази даних, алгоритм переходить до фази вибору та підготовки списку учасників. Модуль жеребкування надсилає запит до таблиці конфігурації турнірів для визначення встановленої системи проведення змагань. Залежно від отриманого значення системи (колова або плей-оф), логіка розгалужується:

- при роботі за системою на виліт (knockout) програма виконує складну вибірку з бази даних, відбираючи ідентифікатори лише тих гравців, які здобули перемогу в попередньому турі (де результат відповідав перемозі білих або чорних фігур відповідно). Учасники, які зазнали поразки, автоматично відсіюються алгоритмом і не потрапляють до фінального списку розподілу pair-list;
- при роботі за коловою системою алгоритм ігнорує проміжні результати та формує повний список, залучаючи абсолютно всіх шахістів, які офіційно зареєстровані як учасники поточного турніру в сполучній таблиці зв'язків.

Після фінального формування списку ідентифікаторів гравців алгоритм виконує перевірку на критичну кількість учасників для створення пар. Якщо кількість шахістів у списку стає меншою за два, система фіксує завершення змагання, сповіщає користувача про визначення фіналіста або повне виконання колового регламенту і припиняє роботу модуля. Коли кількість гравців є

достатньою, запускається механізм випадкового перемішування (рандомізації) елементів списку за допомогою генератора випадкових чисел. Перемішаний масив розділяється навпіл, утворюючи пари опонентів, де один гравець отримує статус білого кольору фігур, а інший – чорного. На фінальному етапі алгоритм відкриває транзакцію до бази даних, послідовно виконує команди додавання нових рядків у таблицю матчів із автоматичним присвоєнням згенерованого номера наступного туру, встановленням нульового прапорця зіграності та фіксацією поточної системної дати. Оскільки робота з розподілом учасників базується на імовірнісних моделях, застосований підхід дозволяє нівелювати суб'єктивні помилки та забезпечити математичну обґрунтованість жеребкування [29]. Після успішного завершення операцій запису інтерфейс форми викликає метод оновлення таблиці результатів, візуалізуючи зміну турнірного стану для користувача.

Розроблене алгоритмічне та архітектурне забезпечення гарантує високу стабільність змагального процесу, повністю автоматизує рутинну роботу шахового арбітра з контролю за регламентом та забезпечує швидке й безпомилкове функціонування системи при будь-яких сценаріях проведення турнірів.

2.5 Висновок до другого розділу

У другому розділі кваліфікаційної роботи було проведено комплексне та детальне проектування архітектури, інформаційної інфраструктури та внутрішньої логіки розроблюваної інформаційної системи. За результатами виконання етапу проектування можна зробити наступні висновки:

1. Сформовано деталізовані вимоги до ПЗ. Визначено вичерпний перелік функціональних вимог, що базується на сучасних інженерних стандартах, серед яких ключовими є управління реєстром гравців, конфігурація змагань, автоматизація жеребкування за різними системами та динамічний розрахунок турнірного становища. Окреслено нефункціональні вимоги, зокрема

гарантування надійності зберігання даних при збоях, висока швидкодія транзакцій та ергономічність графічного інтерфейсу користувача.

2. Розроблено статичні архітектурні моделі. За допомогою мови графічного моделювання UML побудовано діаграми прецедентів (Use Case Diagram) та класів (Class Diagram). Це дозволило чітко розмежувати ролі користувача (спортивного арбітра) в системі та визначити структуру базових програмних об'єктів (Player, Tournament, Match), що забезпечує гнучкість, розширюваність та модульність майбутнього коду мовою C#.

3. Спроектовано реляційну базу даних. На основі принципів нормалізації розроблено ER-схему (Entity-Relationship) у середовищі Microsoft SQL Server. Визначено оптимальну структуру таблиць, первинні та зовнішні ключі, а також типи зв'язків між сутностями (зокрема, впроваджено сполучну таблицю для вирішення зв'язку «багато-до-багатьох»), що гарантує абсолютну цілісність інформації та відсутність надмірності при збереженні турнірних результатів.

4. Змодельовано динамічну поведінку та алгоритмічне забезпечення. Побудовано діаграму послідовності (Sequence Diagram) для візуалізації взаємодії клієнтського додатку з базою даних під час оновлення лідерборду. Крім того, розроблено та алгоритмізовано логіку модуля автоматичного жеребкування у вигляді блок-схеми, яка враховує сувору перевірку завершеності попередніх турів та підтримує алгоритмічне розгалуження для колової системи і системи плей-оф.

5. Обґрунтовано вибір інструментарію. Встановлено, що використання технології Windows Forms у поєднанні з платформою .NET та мовою об'єктно-орієнтованого програмування C# є оптимальним рішенням для реалізації поставлених завдань. Застосування технології ADO.NET для прямого доступу до бази даних забезпечує швидку та надійну роботу десктопного додатка без залучення надлишкових проміжних сервісів.

Таким чином, отримані результати проектування формують стійкий та вичерпний теоретично-практичний фундамент. Розроблена документація, архітектурні схеми та продумані алгоритми дозволяють безпосередньо перейти

до етапу написання вихідного коду, створення графічного інтерфейсу та комплексного тестування готової системи обліку шахових турнірів.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕХНОЛОГІЇ

3.1 Розробка графічного інтерфейсу користувача

Обличчям будь-якої програмної системи є її графічний інтерфейс користувача (GUI). Від його ергономічності, логічної структурованості та інтуїтивної зрозумілості безпосередньо залежить ефективність роботи кінцевого користувача – спортивного арбітра. Для реалізації клієнтської частини розроблюваного додатку було обрано технологію Windows Forms платформи .NET, яка надає багатий набір вбудованих візуальних компонентів для швидкого та надійного проектування десктопних застосунків.

Проектування інтерфейсу здійснювалося за принципом мінімізації когнітивного навантаження на користувача: усі ключові інструменти згруповані за функціональним призначенням, а доступ до основних модулів системи здійснюється з єдиного навігаційного центру – головного вікна програми (дашборду). Такий підхід повністю відповідає сучасним вимогам до проектування людино-машинної взаємодії.

Робота з програмним комплексом «Chess Master Pro v1.0» розпочинається зі стартового вікна (головного меню), яке виконує роль єдиного навігаційного центру програми. Інтерфейс цього вікна спроектовано у мінімалістичному стилі з чіткою візуальною ієрархією, що дозволяє користувачеві миттєво зорієнтуватися в доступних функціях без попереднього навчання. . Зовнішній вигляд головного вікна програми наведено на рисунку 3.1.

У верхній частині форми розташований масивний інформаційний заголовок «УПРАВЛІННЯ ТУРНІРАМИ», який слугує візуальним якорем. Центральну частину вікна займає блок основної навігації, реалізований у вигляді великих, зручних кнопок із графічними піктограмами (іконками):

1. «Управління гравцями» – забезпечує перехід до модуля формування глобального реєстру шахістів, де відбувається додавання нових осіб, редагування їхніх рейтингів та розрядів.

2. «Список турнірів» – відкриває доступ до створення нових змагальних подій, конфігурації систем жеребкування та перегляду історії раніше проведених турнірів.

3. «Вихід» – кнопка завершення сеансу роботи з програмою, яка для запобігання випадковому натисканню виділена контрастним червоним кольором.

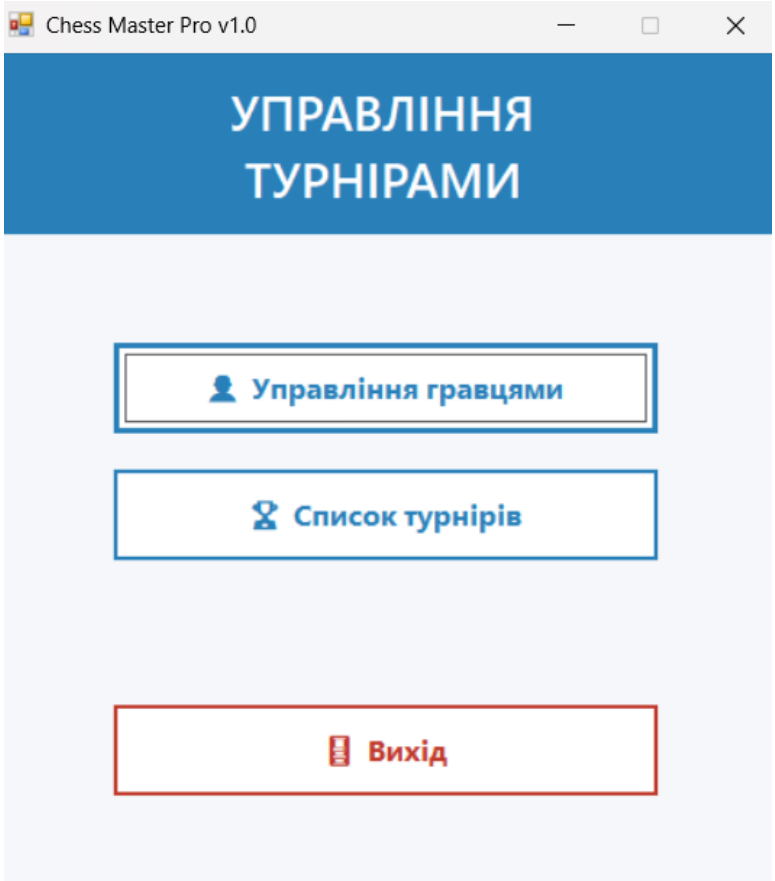


Рисунок 3.1 – Стартове меню системи обліку шахових турнірів

Такий модульний підхід до організації стартового меню дозволяє чітко розмежувати підготовчий етап роботи адміністратора від процесу безпосереднього проведення змагань

Для забезпечення виконання вимог щодо формування бази учасників було розроблено модуль управління реєстром гравців. Перехід до цього модуля

відкриває вікно «Список учасників», інтерфейс якого оптимізовано для швидкої обробки великих масивів даних.

Основну площу робочої області займає інтерактивний табличний компонент DataGridView, у якому відображаються всі збережені в базі даних SQL Server шахісти. Таблиця структурована за ключовими атрибутами: унікальний ідентифікатор (ID), прізвище та ім'я (ПІБ), поточний рейтинг Ело, спортивний розряд та належність до шахового клубу. Під таблицею розташована панель інструментів для маніпуляції даними: кнопки оновлення списку, видалення вибраного запису та додавання нового учасника. Важливим елементом ергономіки є кольорове кодування керуючих елементів (зелений колір для створення записів, червоний контур – для видалення), що інтуїтивно запобігає помилковим діям оператора, забезпечуючи високий рівень ефективності. Інтерфейс модуля роботи з гравцями наведено на рисунку 3.2.

При ініціалізації процесу створення нового запису (натискання кнопки «Додати гравця») система викликає спливаюче модальне діалогове вікно «Картка гравця». Використання модальних форм дозволяє сфокусувати увагу арбітра виключно на процесі введення, що мінімізує ризик внесення некоректних даних. Після натискання кнопки «Зберегти» система здійснює автоматичну валідацію введеної інформації, формує параметризований SQL-запит та відправляє дані на сервер.

Форма реєстрації містить оптимізований набір елементів введення:

- текстові поля (TextBox) для фіксації прізвища, імені та назви шахового клубу;
- компонент числового введення (NumericUpDown) для точного та безпомилкового задання початкового рейтингу;
- випадаючий список (ComboBox) для стандартизованого вибору спортивного розряду (наприклад, «б/р» – без розряду, «КМС» тощо).

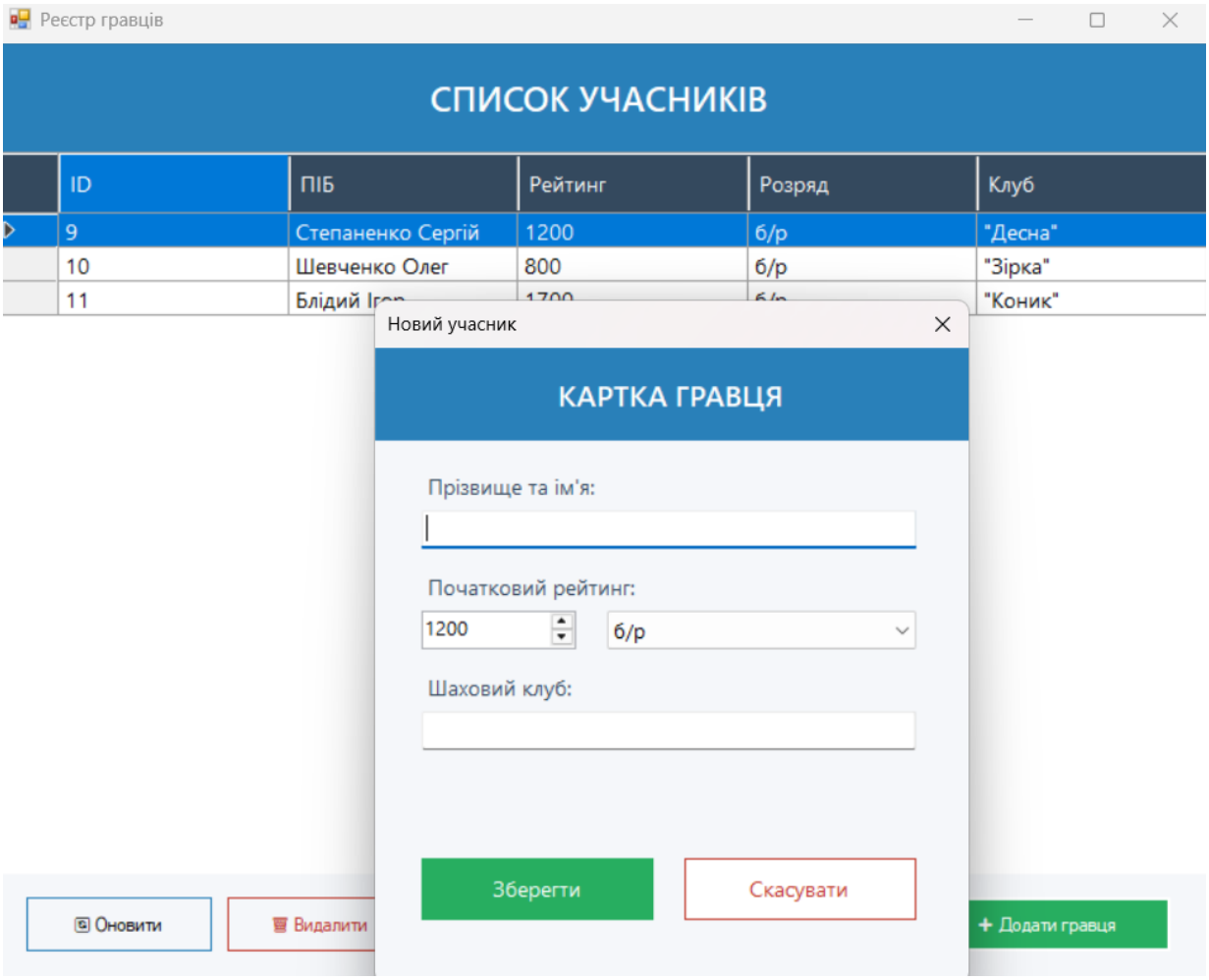


Рисунок 3.2 – Інтерфейс модуля управління реєстром та реєстрації нових гравців

Після натискання кнопки «Зберегти» система здійснює автоматичну валідацію введеної інформації, формує параметризований SQL-запит (INSERT) і відправляє дані на сервер, миттєво оновлюючи відображення у головній таблиці реєстру.

Наступним логічним етапом після формування бази учасників є організація та конфігурація самих змагань. Для цього в системі розроблено модуль «Список турнірів», який забезпечує централізоване управління всіма шаховими подіями.

Головне вікно модуля містить інформаційну таблицю DataGridView, яка відображає історію та поточний стан усіх створених змагань. Кожен запис містить ключові атрибути: унікальний ідентифікатор, назву, систему

проведення, дату початку та поточний статус (наприклад, «Активний»). Панель інструментів під таблицею продовжує загальну концепцію ергономіки та використовує кольорове маркування кнопок: зелена – для створення нового турніру, синя – для переходу до управління обраним змаганням, помаранчева – для зміни його статусу, а червона – для видалення. Таке розділення функцій суттєво зменшує ризик випадкового видалення або зміни даних.

Для ініціалізації нового змагання адміністратор натискає кнопку «Новий турнір», що викликає спливаюче модальне вікно конфігурації. Форма створення спроектована максимально лаконічно і містить наступні елементи введення:

- текстове поле (TextBox) для визначення офіційної назви події (наприклад, для проведення поточних регіональних змагань, таких як Чемпіонат Тернопільської області);
- випадаючий список (ComboBox) для фіксації затвердженої регламентом системи проведення (наприклад, система «На виліт» / плей-оф або колова система);
- компонент вибору дати та часу (DateTimePicker), який надає користувачу зручний інтерактивний графічний календар для безпомилкового встановлення дати початку турніру (наприклад, планування події на 25 червня 2026 р.).

Після заповнення всіх необхідних параметрів натискання кнопки «Створити» ініціює збереження даних на сервері БД. Новий турнір миттєво з'являється у загальному списку та стає доступним для переходу до етапу реєстрації гравців і генерації турів. Графічний інтерфейс процесу конфігурації нового турніру наведено на рисунку 3.3.

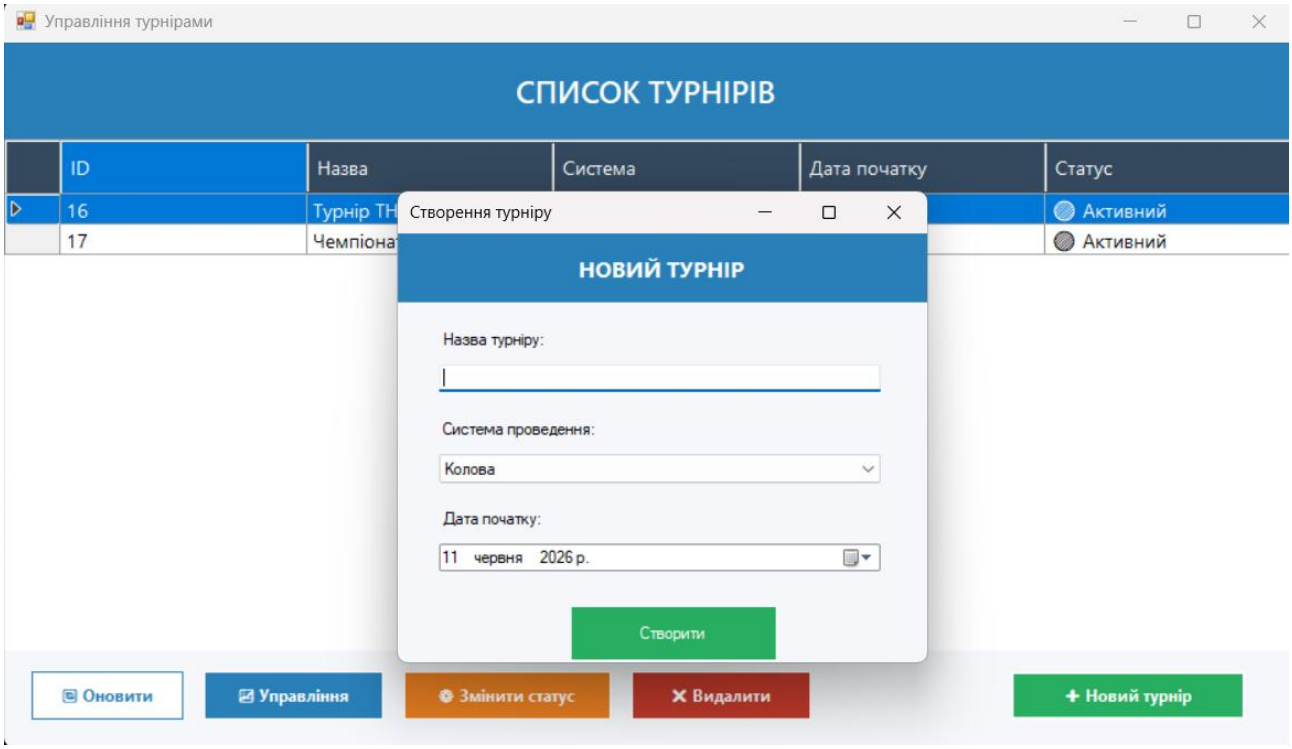


Рисунок 3.3 – Модуль управління змаганнями та вікно конфігурації нового турніру

Після успішної конфігурації нового змагання або вибору існуючого зі списку, користувач переходить до головного робочого простору – панелі управління конкретним турніром. Це найкомплексніший модуль системи, який об'єднує функції реєстрації учасників, ініціалізації турів та динамічного моніторингу результатів.

Інтерфейс вікна управління (на прикладі «Турніру ТНТУ») спроектовано з використанням чіткого візуального зонування, що забезпечує максимальну концентрацію арбітра на поточних завданнях. Екран розділено на три основні функціональні блоки:

1. Зона глобальної вибірки (зліва). Містить компонент DataGridView, який відображає список усіх доступних гравців із бази даних, які ще не зареєстровані на дане змагання.

2. Зона поточного складу (справа зверху). Відображає відібраних учасників поточного турніру.

3. Блок керування (по центру). Включає навігаційні кнопки для переміщення шахістів між списками («Додати» та «Прибрати»). Такий підхід робить процес заявки гравців на турнір візуально прозорим та захищеним від дублювання даних.

Під списком учасників розташовано модуль аналітики – динамічну турнірну таблицю (лідерборд). Вона в режимі реального часу відображає поточні позиції гравців, їхні розряди та кількість набраних очок.

Для виконання бізнес-логіки змагання передбачено окремий блок кольорових кнопок:

- «Сформувати тур» – запускає математичний алгоритм генерації пар опонентів на основі обраної системи жеребкування;
- «Список ігор» – відкриває діалогове вікно для фіксації результатів матчів (перемог, поразок, нічиїх);
- «Оновити бали» – ініціює SQL-запит для перерахунку турнірного становища та оновлення лідерборду після завершення партій.

Таке компонування дозволяє судді контролювати весь хід змагання з єдиного вікна, не перемикаючись між безліччю вкладок. Інтерфейс робочої області турніру наведено на рисунку 3.4.

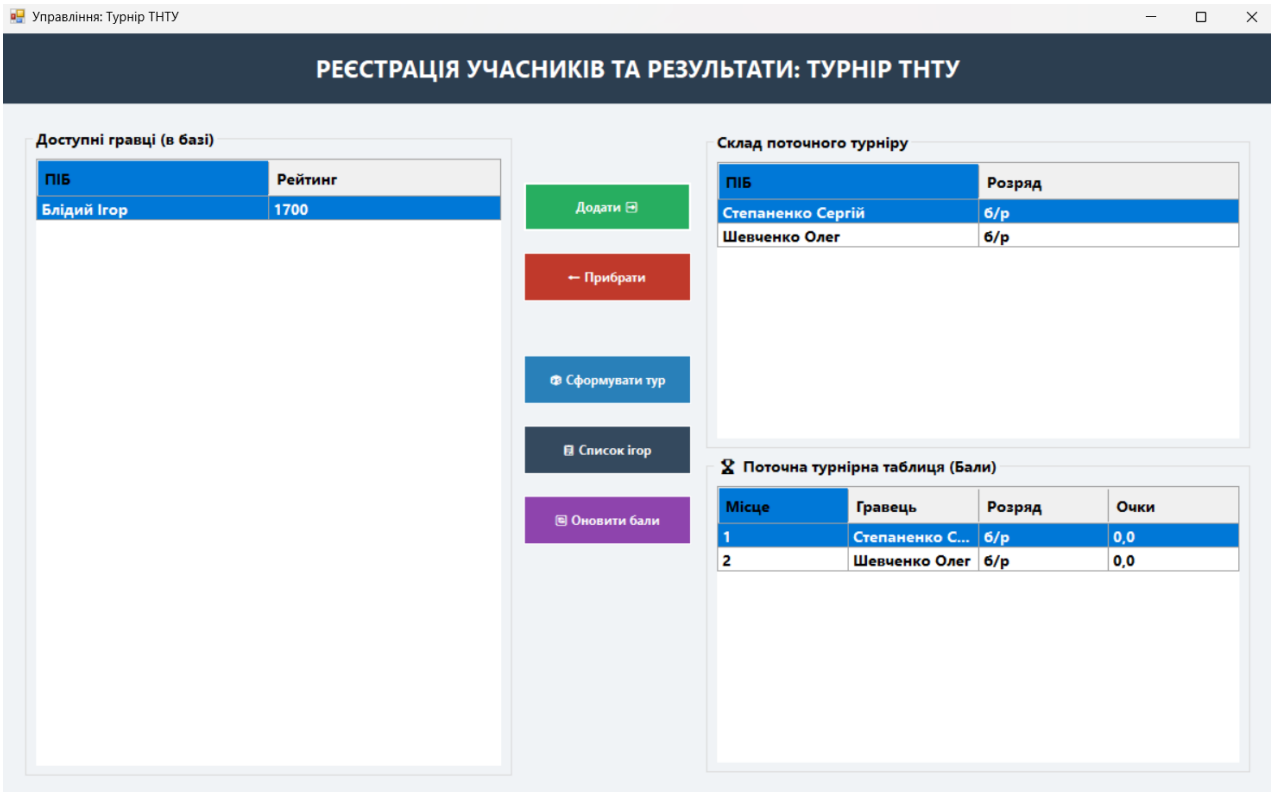


Рисунок 3.4 – Робоча область реєстрації учасників та моніторингу результатів турніру

Важливим критерієм якісного графічного інтерфейсу є своєчасний та зрозумілий зворотний зв'язок системи на дії оператора. Оскільки процес формування ігрових пар вимагає виконання складних математичних алгоритмів та транзакційних звернень до бази даних, система повинна чітко інформувати суддю про результати цих обчислень.

Після ініціалізації жеребкування (натискання кнопки «Сформувати тур») та успішного виконання внутрішньої бізнес-логіки, програма генерує інформаційне модальне вікно за допомогою стандартного компонента MessageBox. Це вікно виконує функцію звіту про виконану операцію. Воно містить відповідну піктограму успіху та текстовий блок, у якому виводяться ключові параметри згенерованого етапу: підтвердження успішності операції, тип застосованої системи жеребкування (наприклад, «Колова»), порядковий номер сформованого туру та загальна кількість створених ігрових матчів.

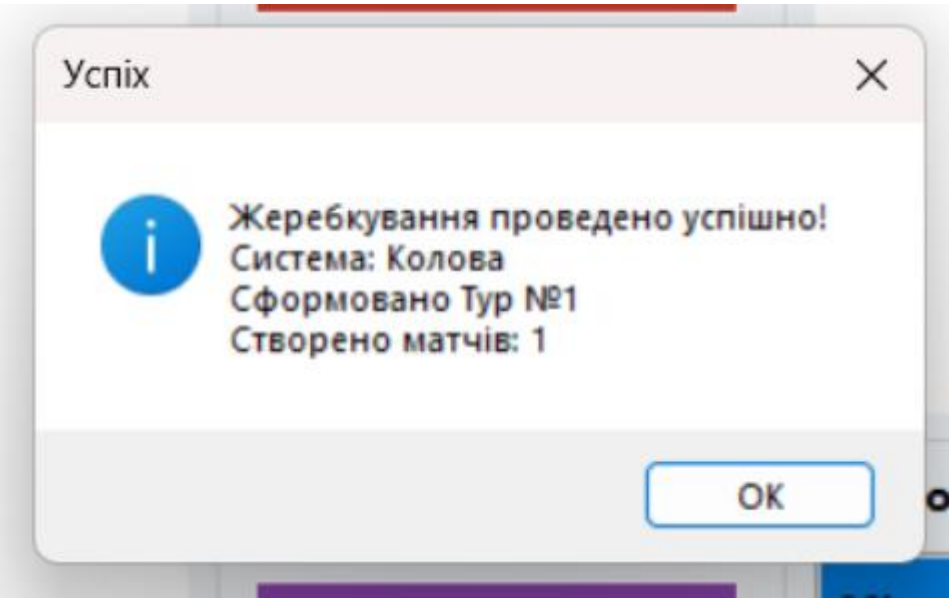


Рисунок 3.5 – Інформаційне вікно результатів автоматичного жеребкування

Для безпосередньої реєстрації підсумків зіграних партій розроблено спеціалізовану форму «Список матчів та результати», яка викликається натисканням кнопки «Список ігор» у робочій області турніру.

У цьому вікні арбітру доступна зведена таблиця поточних ігрових пар із чітким розподілом за кольором фігур (білі та чорні гравці). З метою унеможливлення введення некоректних даних (наприклад, випадкових друкарських помилок при ручному введенні тексту), процес фіксації результату реалізовано через додаткове модальне вікно. Після вибору потрібної партії та натискання кнопки «Встановити результат», користувачеві пропонується вибрати стандартизоване математичне значення з випадаючого списку (ComboBox): 1 (перемога білих), 0 (перемога чорних) або 0.5 (нічия). Інтерфейс модуля введення результатів наведено на рисунку 3.6.

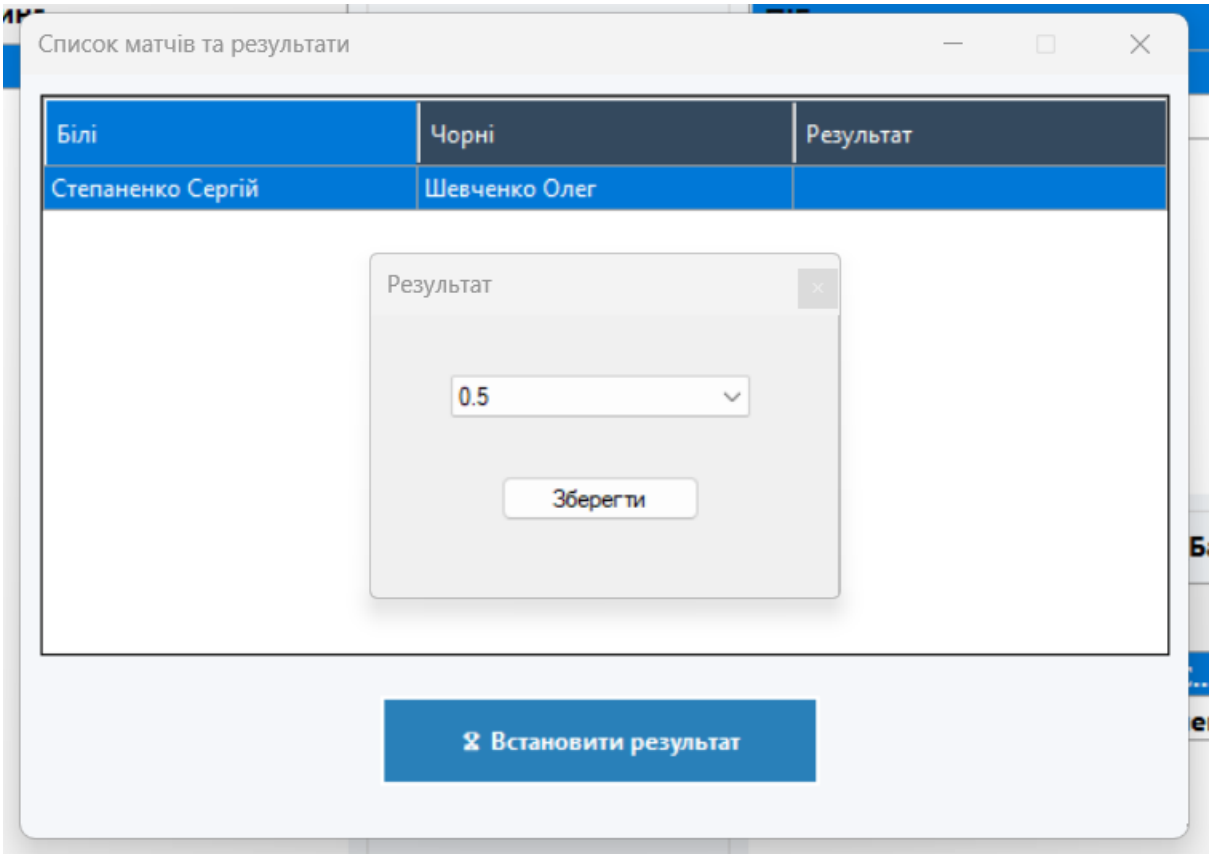


Рисунок 3.6 – Модуль фіксації результатів матчів поточного туру

Після підтвердження вибору кнопкою «Зберегти» система автоматично записує результат у базу даних SQL Server та оновлює статус матчу в таблиці.

3.2 Програмна реалізація бізнес-логіки та взаємодії з базою даних

В основі інформаційної системи лежить комплексна бізнес-логіка, яка забезпечує автоматизацію рутинних процесів проведення шахового турніру. Найбільш ресурсомістким та алгоритмічно складним компонентом клієнтського додатка є модуль автоматичного жеребкування (генерації ігрових пар). Його завдання полягає в динамічному аналізі поточного стану турніру, відборі валідних учасників залежно від обраної системи змагань та випадковому розподілі кольору фігур.

Для реалізації цього функціоналу мовою C# було розроблено метод BtnGenerateMatches_Click, який інкапсульовано всередині класу

TournamentManagementForm. Взаємодія з базою даних реалізована за допомогою технології ADO.NET, зокрема з використанням оптимізованих класів SqlConnection, SqlCommand та SqlDataReader.

Фрагмент програмного коду, що відповідає за перевірку стану турніру, відбір учасників та генерацію ігрових пар, наведено у Лістингу 3.1.

Лістинг 3.1 – Алгоритм генерації ігрових пар та перевірки стану турніру (C#)

```
private void BtnGenerateMatches_Click(object sender, EventArgs e)
{
    string systemType = "";
    List<int> playersToPair = new List<int>();

    using (SqlConnection conn = new
SqlConnection(connectionString))
    {
        conn.Open();

        // 1. Отримання типу системи турніру
        SqlCommand cmd = new SqlCommand("SELECT SystemType FROM
Tournaments WHERE TournamentID = @tid", conn);
        cmd.Parameters.AddWithValue("@tid", tournamentId);
        systemType = cmd.ExecuteScalar()?.ToString() ?? "Колова";

        // 2. Визначення номера наступного туру та валідація
попереднього
        SqlCommand cmdRound = new SqlCommand("SELECT
ISNULL(MAX(RoundNumber), 0) FROM Matches WHERE TournamentID =
@tid", conn);
        cmdRound.Parameters.AddWithValue("@tid", tournamentId);
        int currentMaxRound =
Convert.ToInt32(cmdRound.ExecuteScalar());
        int nextRoundNumber = currentMaxRound + 1;

        if (currentMaxRound > 0)
        {
            SqlCommand cmdCheckUnfinished = new SqlCommand(
                "SELECT COUNT(*) FROM Matches WHERE TournamentID =
@tid AND RoundNumber = @round AND Result IS NULL", conn);
            cmdCheckUnfinished.Parameters.AddWithValue("@tid",
tournamentId);
            cmdCheckUnfinished.Parameters.AddWithValue("@round",
currentMaxRound);

            if
(Convert.ToInt32(cmdCheckUnfinished.ExecuteScalar()) > 0)
            {
```

```
                MessageBox.Show($"Неможливо створити тур
№{nextRoundNumber}. У турі №{currentMaxRound} ще залишилися
незіграні матчі!",
                                "Попередження",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }
        }

        // 3. Відбір гравців залежно від системи турніру (Колова
або На виліт)
        if (nextRoundNumber == 1 || (!systemType.Contains("виліт")
&& !systemType.ToUpper().Contains("KNOCKOUT")))
        {
            foreach (DataGridViewRow row in dgvParticipants.Rows)
            {
                if (row.Cells["PlayerID"].Value != null)

playersToPair.Add(Convert.ToInt32(row.Cells["PlayerID"].Value));
            }
        }
        else
        {
            // Відбір виключно переможців попереднього туру
            string sqlWinners = @"SELECT DISTINCT CASE WHEN Result
= 1 THEN WhitePlayerID
                                                    WHEN Result
= 0 THEN BlackPlayerID ELSE NULL END
                                  FROM Matches WHERE TournamentID
= @tid AND RoundNumber = @prevRound AND Result IS NOT NULL";
            using (SqlCommand cmdWinners = new
SqlCommand(sqlWinners, conn))
            {
                cmdWinners.Parameters.AddWithValue("@tid",
tournamentId);
                cmdWinners.Parameters.AddWithValue("@prevRound",
currentMaxRound);
                using (SqlDataReader reader =
cmdWinners.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        if (reader[0] != DBNull.Value)
playersToPair.Add(Convert.ToInt32(reader[0]));
                    }
                }
            }
        }

        if (playersToPair.Count < 2)
        {
```

```
            MessageBox.Show("Турнір завершено або залишився один
фіналіст!",          "Інформація",          MessageBoxButtons.OK,
MessageBoxIcon.Information);
            return;
        }

        // 4. Рандомізація та розподіл кольорів (Білі/Чорні)
        Random rng = new Random();
        var     shuffled     =     playersToPair.OrderBy(a     =>
rng.Next()).ToList();

        string sqlInsert = "INSERT INTO Matches (TournamentID,
WhitePlayerID,  BlackPlayerID,  RoundNumber,  Result,  IsPlayed,
MatchDate) " +
                           "VALUES (@tid, @white, @black, @round,
NULL, 0, GETDATE())";

        for (int i = 0; i < (shuffled.Count / 2) * 2; i += 2)
        {
            using      (SqlCommand      cmdInsert      =      new
SqlCommand(sqlInsert, conn))
            {
                cmdInsert.Parameters.AddWithValue("@tid",
tournamentId);
                cmdInsert.Parameters.AddWithValue("@white",
shuffled[i]);
                cmdInsert.Parameters.AddWithValue("@black",
shuffled[i + 1]);
                cmdInsert.Parameters.AddWithValue("@round",
nextRoundNumber);
                cmdInsert.ExecuteNonQuery();
            }
        }
        LoadLeaderboard();    //    Синхронне    оновлення    таблиці
результатів
    }
}
```

Представлений алгоритм функціонує за принципом каскадної обробки даних і містить кілька критично важливих інженерних рішень:

1. Валідація станів (Guard Clauses). Програма виконує агрегуючий SQL-запит COUNT(*) для пошуку записів із порожнім результатом (Result IS NULL). Якщо виявлено хоча б одну незавершену партію, виконання методу негайно переривається, що унеможливлює порушення хронології турніру.

2. Алгоритмічне розгалуження систем. Реалізовано гнучку вибірку учасників за допомогою методів об’єктно-орієнтованої ідентифікації станів

системи. Для колової системи алгоритм використовує повний пул гравців із локальної колекції (зчитуючи їх з DataGridView), що зменшує навантаження на базу даних. Натомість для системи плей-оф виконується комплексний SQL-запит із використанням умовного оператора CASE, який динамічно визначає ідентифікатори гравців (WhitePlayerID або BlackPlayerID), які здобули перемогу у попередньому раунді.

3. Безпечна рандомізація. Перемішування списку учасників реалізовано за допомогою методів розширення LINQ (OrderBy(a => rng.Next())), що забезпечує компактність коду, гарантуючи при цьому високий рівень випадковості при призначенні кольору фігур (парні індекси масиву отримують білі фігури, непарні – чорні).

4. Запобігання SQL-ін'єкціям. Усі взаємодії з реляційною базою даних здійснюються виключно через параметризовані запити (використання колекції Parameters.AddWithValue), що гарантує абсолютну безпеку та стабільність обробки типів даних на стороні СУБД.

Після успішної генерації пар система здійснює запис інформації у таблицю Matches та автоматично викликає метод LoadLeaderboard(), який оновлює інтерфейс користувача. Використаний архітектурний підхід забезпечує модульність системи, що відповідає сучасним принципам розробки програмного забезпечення, та гарантує коректність виконання транзакційних операцій згідно з принципами ACID.

3.3 Тестування інформаційної системи

Завершальним етапом життєвого циклу розробки програмного забезпечення є комплексне тестування створеного продукту. Метою цього етапу є перевірка відповідності реалізованого функціоналу початковим вимогам системи, виявлення та усунення можливих програмних дефектів, а також перевірка механізмів захисту від некоректних дій оператора, що в інженерній практиці називається захистом від помилок користувача.

Враховуючи архітектуру розробленого десктопного додатка, для перевірки його працездатності було застосовано метод ручного функціонального тестування (Black-box testing) та тестування графічного інтерфейсу користувача (GUI Testing) , що є загальноприйнятою практикою в інженерії ПЗ. Особливу увагу під час випробувань було приділено валідації вхідних даних та реакції системи на навмисне порушення логіки проведення змагань.

Перша серія тестових сценаріїв була спрямована на перевірку модуля реєстрації гравців та управління глобальним реєстром шахістів. Під час випробувань моделювалися ситуації, коли користувач намагався зберегти профіль нового учасника з порожнім текстовим полем для введення прізвища та імені або із некоректно заданим початковим рейтингом. Завдяки впровадженим механізмам попередньої валідації на рівні інтерфейсу, клієнтська програма успішно перехоплювала такі спроби ще до моменту формування транзакції. Блокування некоректного SQL-запиту супроводжувалося викликом модального вікна з попередженням, що повністю виключило ризик засмічення бази даних порожніми або завідомо невалідними записами.

Другий етап випробувань стосувався перевірки надійності ядра бізнес-логіки – алгоритму автоматичного жеребкування та генерації турів. Критично важливим тестом була спроба користувача натиснути кнопку «Сформувати тур» у робочій області «Турніру ТНТУ» за умови, що у поточному раунді ще залишалися ігри без внесених результатів. Під час тестування програма виконувала агрегуючий запит до бази даних, виявляла наявність записів із неозначеним результатом (Result IS NULL) і миттєво переривала хід виконання методу BtnGenerateMatches_Click. Оператору виводилося чітке попередження про неможливість генерації наступного етапу до моменту завершення всіх поточних партій. Це підтвердило ефективність роботи захисних умов (Guard clauses) для збереження суворої хронології змагань.

Окремо було проведено тестування математичної коректності формування ігрових пар для різних типів змагальних систем. При виборі

системи «На виліт» (Knockout) система під час генерації другого та наступних турів виконувала комплексний SQL-запит із умовним оператором CASE, який відсіював переможених шахістів. Тестування підтвердило, що в пул для випадкового перемішування методами LINQ потрапляли виключно ті ідентифікатори гравців, які здобули перемогу (отримали результат 1.0). У випадку, коли кількість таких гравців ставала меншою за двох, алгоритм коректно зупиняв роботу, інформуючи арбітра про успішне фінальне завершення турнірної сітки або визначення єдиного переможця.

Заключний блок тестів оцінював інтеграційну взаємодію клієнтської частини із сервером баз даних Microsoft SQL Server та швидкість реагування інтерфейсу. Тестування сценарію внесення результатів матчів (через випадаючі списки із фіксованими значеннями 1, 0 чи 0.5) показало, що після збереження даних та автоматичного виклику методу LoadLeaderboard() відбувається миттєвий перерахунок загального турнірного становища. Складний аналітичний SQL-запит із групуванням та віконними функціями сортування успішно оновлював компонент DataGridView, розташований у правій частині форми, без візуальних затримок.

Загалом результати комплексного функціонального тестування продемонстрували високий рівень стабільності та відмовостійкості розробленого програмного забезпечення. Інформаційна система демонструє повну відповідність усім заявленим функціональним вимогам, ефективно запобігає втраті чи спотворенню інформації та забезпечує комфортні, захищені умови для автоматизації роботи шахового арбітра [30].

3.4 Висновки до третього розділу

У третьому розділі кваліфікаційної роботи було здійснено повну реалізацію програмної системи для обліку шахових турнірів та проведено її комплексне тестування. Цей етап дозволив перетворити розроблені на попередніх стадіях теоретичні моделі та архітектурні рішення у повноцінно

функціонуючий програмний продукт. За результатами виконання практичної частини можна зробити наступні висновки:

1. Реалізовано ергономічний клієнтський додаток. За допомогою технології Windows Forms розроблено зручний графічний інтерфейс користувача. Використання сучасних компонентів візуалізації (DataGridView, модальних вікон, випадаючих списків, кольорового кодування кнопок) та поділ робочого простору на логічні зони забезпечують інтуїтивне та швидке управління змаганнями. Окрему увагу приділено механізмам зворотного зв'язку, завдяки яким оператор своєчасно отримує візуальні повідомлення про успішність виконання транзакцій чи помилки введення.

2. Запрограмовано стійку бізнес-логіку. Мовою C# успішно реалізовано ядро системи – алгоритм автоматичного жеребкування. Завдяки застосуванню математичної рандомізації засобами LINQ та динамічних умовних конструкцій, система здатна безпомилково формувати ігрові пари як для колової системи, так і для турнірів на виліт. Програмна логіка коректно обробляє специфічні сценарії, зокрема відсів переможених учасників на кожному етапі плей-оф, суворо дотримуючись спортивного регламенту.

3. Забезпечено надійну взаємодію з БД. Впровадження технології ADO.NET (SqlConnection, SqlCommand) дозволило налагодити швидкий транзакційний обмін даними між клієнтським додатком та сервером баз даних. Це гарантує миттєвий перерахунок турнірних балів та цілісність інформації при будь-яких змінах складу учасників. Крім того, використання виключно параметризованих SQL-запитів забезпечило високий рівень безпеки та захист системи від потенційних вразливостей, таких як SQL-ін'єкції, та гарантувало цілісність інформації при оновленні лідербордів.

4. Проведено успішне тестування. Виконання серії функціональних тестів (Black-box testing) та перевірка графічного інтерфейсу (GUI-тестування) підтвердили стабільність програми. Реалізовані механізми перевірки станів (Guard clauses) ефективно захищають базу даних від введення некоректної інформації та унеможливлюють порушення хронологічного порядку

проведення турнірних раундів. Система продемонструвала високу відмовостійкість до непередбачуваних дій користувача.

Створений програмний продукт повністю відповідає поставленому технічному завданню, функціонує без критичних збоїв і готовий до впровадження у реальну практику для автоматизації роботи спортивних шахових суддів та організаторів змагань. Архітектура розробленого додатка є гнучкою та модульною, що відкриває широкі можливості для його подальшого масштабування, зокрема шляхом розширення аналітичного функціоналу, інтеграції розширеної статистики гравців та додавання модулів експорту турнірних звітів у зовнішні формати.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Психологічні причини нещасних випадків і травматизму

Проблема забезпечення безпеки життєдіяльності та охорони праці на сучасному етапі розвитку суспільства є надзвичайно актуальною. Незважаючи на стрімкий розвиток технологій, автоматизацію виробничих процесів та впровадження новітніх систем захисту, рівень травматизму на робочих місцях залишається вагомою проблемою [31]. Статистичні дані та результати розслідувань нещасних випадків свідчать про те, що від 60% до 80% аварій та травм відбуваються не через технічні несправності обладнання, а внаслідок дії так званого «людського фактора» [32]. Саме тому вивчення психологічних причин нещасних випадків є ключовим аспектом у системі профілактики виробничого травматизму.

Психологія безпеки праці розглядає нещасний випадок не як раптову чи випадкову подію, а як закономірний наслідок порушення психологічної надійності працівника або невідповідності його психофізіологічних характеристик вимогам професійної діяльності [33]. Усі психологічні причини, що призводять до створення небезпечних ситуацій та травматизму, в академічній літературі прийнято класифікувати на три великі групи: порушення мотиваційної, орієнтовної та виконавчої частин трудової діяльності.

Порушення мотиваційної частини дій. До цієї групи належать причини, пов'язані з небажанням працівника виконувати дії безпечно. Людина усвідомлює небезпеку, має необхідні знання та навички, але свідомо ігнорує правила [34]. До основних психологічних факторів цієї групи належать:

- схильність до невиправданого ризику. Деякі працівники розглядають ризик як прояв сміливості або спосіб самоствердження в колективі;

- економія часу та зусиль. Нехтування засобами індивідуального захисту або правилами експлуатації обладнання (наприклад, зняття захисних екранів) заради прискорення виконання завдання;
- звикання до небезпеки (ілюзія безпеки). При тривалій роботі в умовах потенційної небезпеки без негативних наслідків у людини притупляється почуття страху. Працівник починає вважати, що його досвід є абсолютною гарантією безпеки, що призводить до втрати пильності та грубих помилок;
- безвідповідальність та недисциплінованість. Відсутність внутрішньої мотивації до безпечної праці, що часто є наслідком низької корпоративної культури безпеки на підприємстві.

Порушення орієнтовної частини дій. Ця група причин пов'язана з неможливістю працівника правильно оцінити ситуацію та прийняти адекватне рішення. Це виникає тоді, коли людина не помічає небезпеки або не розуміє її реального рівня. Психологічні причини тут включають:

- стан втоми та перевтоми. Тривала розумова або фізична напруга (що особливо актуально для ІТ-фахівців, розробників ПЗ та операторів ЕОМ) призводить до виснаження нервової системи. Знижується концентрація уваги, погіршується пам'ять, збільшується час реакції на зовнішні подразники;
- монотонія. Виконання одноманітних, рутинних операцій в умовах дефіциту сенсорної інформації викликає у працівника стан апатії, сонливості та зниження рівня пильності. Людина починає виконувати дії автоматично, не фіксуючи змін у навколишньому середовищі;
- Еемоційний стрес. Сильні емоційні переживання (конфлікти на роботі, сімейні проблеми, страх звільнення) блокують аналітичні центри кори головного мозку. У стані афекту або сильного стресу людина втрачає здатність до раціонального мислення, звужується обсяг уваги («тунельне бачення»), що призводить до прийняття хибних рішень.

Порушення виконавчої частини дій. Такі порушення виникають тоді, коли працівник хоче виконати дію правильно, усвідомлює небезпеку, але через

певні психофізіологічні особливості або тимчасові стани не може цього зробити. До них належать:

- невідповідність психофізіологічних характеристик. Недостатня швидкість реакції, слабка координація рухів, дефекти зору чи слуху, які не дозволяють людині безпечно працювати зі складною або небезпечною технікою;
- вплив психотропних речовин. Вживання алкоголю, наркотичних засобів або навіть певних медичних препаратів (наприклад, сильних заспокійливих чи антигістамінних) кардинально змінює психічний стан людини. Знижується самоконтроль, спотворюється сприйняття реальності та уповільнюється моторна реакція;
- паніка. У разі виникнення позаштатної (аварійної) ситуації непідготовлений працівник може впасти у стан паніки. Це супроводжується хаотичними, нецілеспрямованими рухами, ступором або навпаки – неконтрольованою втечею, що часто призводить до травмування як самої людини, так і її колег.

Важливу роль у формуванні психологічних причин травматизму відіграє також соціально-психологічний клімат у колективі. Токсична атмосфера, часті конфлікти, авторитарний стиль управління та постійний психологічний тиск з боку керівництва (мобінг) створюють перманентний фоновий стрес. Працівник, який постійно перебуває в стані тривоги, робить у кілька разів більше помилок, ніж той, хто працює в умовах психологічного комфорту.

Враховуючи вищезазначене, для ефективного запобігання виробничому травматизму недостатньо лише вдосконалювати техніку. Сучасна інженерія безпеки вимагає впровадження заходів психологічної профілактики. До них належать: професійний психологічний відбір (особливо для робіт підвищеної небезпеки), оптимізація режимів праці та відпочинку для зняття втоми і монотонії, проведення соціально-психологічних тренінгів для підвищення стійкості до стресу, а також формування високої корпоративної культури, де безпека є головною цінністю кожного члена колективу. Лише комплексний

підхід, що враховує психологічну природу людських помилок, здатний мінімізувати рівень травматизму та зберегти життя і здоров'я працівників.

4.2 Соціальне значення охорони праці

Охорона праці є не просто набором технічних правил, санітарних норм чи юридичних інструкцій. Це складна багатогранна система, яка має глибокий вплив на всі сфери функціонування держави та суспільства. Соціальне значення охорони праці визначається її роллю у збереженні найвищої соціальної цінності – життя та здоров’я людини в процесі її трудової діяльності [35]. Згідно зі статтею 43 Конституції України, кожен має право на належні, безпечні і здорові умови праці. Забезпечення цього права є одним із головних критеріїв розвитку цивілізованого, демократичного суспільства.

Соціальна значущість охорони праці проявляється у кількох взаємопов'язаних аспектах: демографічному, гуманістичному, соціально-економічному та психологічному. Кожен із цих напрямків безпосередньо впливає на якість життя нації та стабільність держави.

Демографічний аспект. Трудовий потенціал є основою економічної могутності будь-якої держави. Охорона праці відіграє ключову роль у збереженні генофонду нації. Створення безпечних і нешкідливих умов на робочих місцях безпосередньо сприяє зниженню рівня виробничого травматизму, професійної захворюваності та смертності громадян працездатного віку. Особливе соціальне значення має охорона праці жінок та молоді. Звільнення жінок від важких фізичних робіт та впливу токсичних речовин є запорукою народження здорового покоління, а захист молоді гарантує гармонійний фізичний та психологічний розвиток майбутніх фахівців. Таким чином, охорона праці є інструментом боротьби з демографічною кризою та передчасною втратою трудових ресурсів.

Гуманістичний аспект. У центрі сучасної системи охорони праці знаходиться особистість. Гуманізація праці означає перехід від ставлення до

людини як до «гвинтика» виробничого механізму до визнання її головною цінністю. Умови, в яких людина працює (а це близько третини всього свідомого життя), формують її особистість, світогляд та соціальну поведінку. Безпечні, ергономічні та естетично привабливі робочі місця (що є стандартом у сучасних ІТ-компаніях) не лише запобігають травмам, а й викликають у працівника відчуття задоволення від своєї діяльності. Робота перестає бути важким тягарем і стає засобом самореалізації. Працівник, чиї права на безпечну працю поважаються, відчуває свою соціальну значущість, що сприяє підвищенню рівня його загальної культури та інтелектуального розвитку.

Соціально-економічний аспект. Соціальне та економічне значення охорони праці нерозривно пов'язані. Нещасні випадки та професійні захворювання лягають важким фінансовим тягарем не лише на підприємства, а й на суспільство в цілому. Зниження рівня травматизму дозволяє суспільству зберігати колосальні кошти, які інакше були б витрачені на:

- оплату листків непрацездатності (лікарняних);
- медичну та соціальну реабілітацію постраждалих;
- виплати пенсій по інвалідності та допомоги у разі втрати годувальника;
- підготовку нових кадрів на заміну травмованим працівникам.

Заощаджені завдяки ефективній охороні праці державні кошти можуть бути спрямовані на реалізацію інших важливих соціальних програм: розвиток медицини, освіти, будівництво інфраструктури, що в підсумку підвищує загальний рівень добробуту всього населення. Крім того, комфортні та безпечні умови праці сприяють зростанню продуктивності праці на 15–20%, підвищують якість виробленої продукції та зменшують плинність кадрів.

Соціально-психологічний аспект. Стан охорони праці має безпосередній вплив на соціальну стабільність і психологічний клімат як у рамках окремого колективу, так і суспільства загалом. Людина, яка працює в умовах постійного ризику для життя чи здоров'я, перебуває у стані хронічного стресу. Це напруження переноситься в її сім'ю, провокує конфлікти, збільшує ризик розлучень та розвитку депресивних станів. Травмування або загибель

працівника є непоправною психологічною трагедією для його рідних, яка часто призводить до маргіналізації родини через втрату головного джерела доходів. Натомість, високий рівень охорони праці дає працівникам упевненість у завтрашньому дні, почуття захищеності та стабільності.

Окрім цього, в сучасному світі охорона праці стала важливим елементом корпоративної соціальної відповідальності (КСВ). Підприємства, які інвестують у безпеку та здоров'я своїх співробітників, формують позитивний імідж на ринку праці, що дозволяє їм залучати найкращих фахівців та здобувати довіру партнерів.

Підсумовуючи, можна стверджувати, що соціальне значення охорони праці полягає у формуванні здорового, стабільного та економічно успішного суспільства. Інвестиції у безпеку життєдіяльності та охорону праці є найгуманнішими та найефективнішими інвестиціями, оскільки вони спрямовані на збереження найдорожчого капіталу держави – людського життя.

4.3 Висновки до четвертого розділу

У четвертому розділі кваліфікаційної роботи було досліджено ключові аспекти безпеки життєдіяльності та охорони праці з акцентом на психологічні фактори травматизму та соціальну роль безпеки на виробництві. За результатами аналізу можна зробити наступні висновки:

1. Визначено роль людського фактора. Встановлено, що переважна більшість нещасних випадків виникає не через технічні несправності, а внаслідок психологічних причин: порушень мотиваційної (невиправданий ризик, ігнорування правил), орієнтовної (перевтома, монотонія, стрес) та виконавчої (фізіологічна невідповідність) частин трудової діяльності працівника.

2. Обґрунтовано необхідність психологічної профілактики. Для фахівців, чия діяльність пов'язана з високим інтелектуальним навантаженням та тривалою роботою за комп'ютером (зокрема інженерів-програмістів та

операторів інформаційних систем), критично важливо оптимізувати режими праці і відпочинку, уникати емоційного виснаження та створювати сприятливий соціально-психологічний клімат у колективі.

3. Розкрито багатогранне соціальне значення охорони праці. Доведено, що забезпечення безпечних умов праці виходить за межі вузьких технічних вимог. Воно має потужний демографічний вплив (збереження працездатного населення), економічний ефект (зниження витрат на лікування та компенсації, підвищення продуктивності праці) та сприяє загальній гуманізації суспільства.

4. Підтверджено пріоритетність безпеки. Охорона праці є невід'ємною складовою успішної професійної діяльності. Суворе дотримання норм безпеки життєдіяльності, ергономіки робочого місця та врахування психофізіологічних особливостей людини гарантує не лише збереження здоров'я фахівця, а й високу ефективність виконання ним виробничих завдань.

Підсумовуючи вищезазначене, варто наголосити, що комплексний підхід до питань охорони праці та безпеки життєдіяльності є фундаментальною основою діяльності будь-якого сучасного фахівця. Впровадження розглянутих принципів психологічної профілактики, ергономіки та соціального захисту дозволяє не лише звести до мінімуму виробничі ризики, але й створити оптимальні умови для безпечної, комфортної та високопродуктивної роботи розробника програмного забезпечення.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне завдання щодо автоматизації процесів організації, проведення та обліку результатів шахових турнірів шляхом розробки спеціалізованої інформаційної системи. За результатами виконання роботи можна зробити наступні висновки:

1. Проаналізовано предметну область. Дослідження специфіки проведення спортивних змагань із шахів показало, що ручний облік результатів та жеребкування вимагають значних витрат часу арбітра і містять високий ризик людської помилки. На основі аналізу сформовано детальні функціональні та нефункціональні вимоги до програмного продукту.

2. Спроектовано архітектуру та базу даних. Розроблено логічну та фізичну моделі даних, що дозволило чітко розмежувати програмні об'єкти. На базі реляційної СУБД Microsoft SQL Server створено оптимізовану базу даних із використанням зовнішніх ключів, що гарантує цілісність, надійність збереження та відсутність дублювання інформації про реєстр шахістів, конфігурації турнірів та зіграні матчі.

3. Реалізовано програмну бізнес-логіку та інтерфейс. Засобами мови об'єктно-орієнтованого програмування C# та платформи .NET (Windows Forms) розроблено десктопний клієнтський додаток «Chess Master Pro v1.0». Створено ергономічний графічний інтерфейс із захистом від помилкового введення. Успішно запрограмовано складний алгоритм автоматичного жеребкування для колової системи та турнірів на виліт (плей-оф), який базується на математичному перемішуванні списків і транзакційному виконанні SQL-запитів.

4. Реалізовано модуль аналітики. Створено механізм динамічного підрахунку результатів турніру (лідерборд), який за допомогою складних агрегуючих SQL-запитів у режимі реального часу перераховує бали, розподіляє місця та виводить турнірне становище на екран судді.

5. Проведено комплексне тестування. Виконання серії функціональних тестів (Black-box testing) підтвердило стабільність програми. Система ефективно захищає базу даних від введення некоректної інформації, перехоплює виняткові ситуації та унеможливлює порушення хронологічного порядку проведення турнірних раундів.

6. Опрацьовано питання охорони праці. Досліджено психологічні причини нещасних випадків і травматизму на виробництві та обґрунтовано соціальне значення охорони праці, що є необхідною умовою безпечної професійної діяльності інженера-програміста та оператора інформаційної системи.

Створений програмний продукт повністю відповідає поставленому технічному завданню. Мета кваліфікаційної роботи досягнута. Розроблена програмна система функціонує без збоїв, значно оптимізує роботу спортивного арбітра і є повністю готовою до практичного впровадження у діяльність шахових клубів та організаторів змагань

ПЕРЕЛІК ДЖЕРЕЛ

1. FIDE Laws of Chess. URL: https://www.fide.com/ (2026).
2. Date C. J. Database Systems: A Practical Approach. Pearson, 2019. 960 p.
3. FIDE Rating Regulations. URL: https://ratings.fide.com/ (2026).
4. FIDE Swiss-System Tournament Rules. URL: http://www.fide.com/ (2026).
5. Elo A. The Rating of Chessplayers, Past and Present. Arco, 1978. 206 p.
6. Бабак В.П., Марченко Б.Г., Фриз М.Є. Теорія ймовірностей, випадкові процеси та математична статистика. К.: Техніка, 2004. 288 с.
7. Фриз М.Є., Млинко Б.Б. Умовні лінійні випадкові процеси з дискретним часом // Вісник Хмельницького нац. ун-ту. 2022. № 3. С. 7–12.
8. Шаховий кодекс України. Київ : Мінмолодьспорт, 2021. 110 с.
9. Pressman R. S. Software Engineering: A Practitioner's Approach. McGraw-Hill, 2020. 840 p.
10. Sommerville I. Software Engineering (10th ed.). Pearson, 2016. 792 p.
11. Microsoft WinForms Docs. URL: https://learn.microsoft.com/ (2026).
12. Krug S. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability. New Riders, 2014. 216 p.
13. Silberschatz A. Database System Concepts. McGraw-Hill, 2020. 944 p.
14. Nielsen J. Usability Engineering. Academic Press, 1993. 362 p.
15. Babak V. et al. Noise signals: Modelling and Analyses. Cham: Springer Nature, 2025. 222 p.
16. Connolly T. M., Begg C. E. Database Systems: A Practical Approach. Pearson, 2015. 1360 p.
17. SQL Server Tutorials. URL: https://www.sqlservertutorial.net/ (2026).
18. Skeet J. C# in Depth (4th ed.). Manning Publications, 2019. 592 p.
19. Gamma E. et al. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994. 395 p.

20. Бабак В.П., Куц Ю.В., Мислович М.В., Фриз М.Є., Щербак Л.М. Об’єктно-орієнтована ідентифікація стохастичних шумових сигналів. Київ: Наукова думка, 2024. 240 с. https://doi.org/10.15407/978-966-00-1883-9.

21. Fryz M., Mlynko B. Determination of the characteristic function... // Scientific Journal of TNTU. 2023. Vol. 109. P. 16–23.

22. Richter J. CLR via C# (4th ed.). Microsoft Press, 2012. 896 p.

23. Microsoft SQL Server Documentation. URL: https://learn.microsoft.com/sql/ (2026).

24. Troelsen A. C# 12 and .NET 8. Apress, 2024. 800 p.

25. Fowler M. Refactoring: Improving the Design of Existing Code. Addison-Wesley, 2018. 448 p.

26. Microsoft Learn. C# and .NET Documentation. URL: https://learn.microsoft.com/ (2026).

27. Codd E. F. A Relational Model of Data for Large Shared Data Banks. ACM, 1970.

28. Date C. J. SQL and Relational Theory. O'Reilly Media, 2012. 400 p.

29. Fryz M., Mlynko B. Property analysis of multivariate conditional linear random processes // Technol. Audit Prod. Reserv. 2022. Vol. 3, № 2. P. 29–32.

30. Робоче місце користувача ПК: ергономічні вимоги. Київ : Вища школа, 2021. 140 с.

31. Желібо Є.П. Безпека життєдіяльності : підручник / В. В. Зацарний. Київ : Каравела, 2023. 344 с.

32. Яворський В. В. Охорона праці в галузі ІТ. Тернопіль : ТНТУ, 2020. 215 с.

33. ДСТУ 7239:2011. Система стандартів безпеки праці.

34. Гігієнічна класифікація праці. Київ : МОЗ України, 2019. 55 с.

35. Жидецький В.Ц. Охорона праці користувачів комп’ютерів : підручник. Львів : Афіша, 2020. 176 с.

ДОДАТКИ

Додаток А

Лістинг програмного коду модуля управління турнірами

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace ChessTournamentManager
{
    public partial class TournamentManagementForm : Form
    {
        private int tournamentId;
        private string tournamentName;

        private DataGridView dgvAllPlayers;
        private DataGridView dgvParticipants;
        private DataGridView dgvLeaderboard; // Нова таблиця результатів
        private Button btnAddToTournament;
        private Button btnRemoveFromTournament;
        private Button btnGenerateMatches;
        private Button btnShowMatches;
        private Button btnRefreshLeaderboard; // Кнопка для оновлення балів

        // Рядок підключення до БД (залишено твій сервер та базу)
        string connectionString = @"Server=LENOVO-IDEAPAD-\SQLEXPRESS; Database=ChessTournamentDB; Integrated Security=True; TrustServerCertificate=True;";

        public TournamentManagementForm(int id, string name)
        {
            this.tournamentId = id;
            this.tournamentName = name;
            InitializeCustomDesign();
            LoadData();
            LoadLeaderboard(); // Завантажуємо бали при старті форми
        }

        private void InitializeCustomDesign()
        {
            this.Text = $"Управління: {tournamentName}";
            this.Size = new Size(1200, 750); // Збільшено форму для ідеального розміщення лідерборду
            this.StartPosition = FormStartPosition.CenterScreen;
            this.BackColor = Color.FromArgb(240, 243, 246);
```

```
            // 1. Верхній заголовок
            Label lblHeader = new Label
            {
                Text = $"РЕЄСТРАЦІЯ УЧАСНИКІВ ТА РЕЗУЛЬТАТИ: 
{tournamentName.ToUpper()}",
                Dock = DockStyle.Top,
                Height = 65,
                TextAlign = ContentAlignment.MiddleCenter,
                Font = new Font("Segoe UI", 16, FontStyle.Bold),
                BackColor = Color.FromArgb(44, 62, 80),
                ForeColor = Color.White
            };

            // 2. Головна сітка (3 колонки: 40% - 15% - 45%)
            TableLayoutPanel mainLayout = new TableLayoutPanel
            {
                Dock = DockStyle.Fill,
                ColumnCount = 3,
                RowCount = 1,
                Padding = new Padding(20)
            };
            mainLayout.ColumnStyles.Add(new 
ColumnStyle(SizeType.Percent, 40.0f));
            mainLayout.ColumnStyles.Add(new 
ColumnStyle(SizeType.Percent, 15.0f));
            mainLayout.ColumnStyles.Add(new 
ColumnStyle(SizeType.Percent, 45.0f));

            // Ліва колонка: Доступні гравці
            GroupBox gbAll = CreateStyledGroupBox("Доступні гравці 
(в базі)");
            dgvAllPlayers = CreateStyledDataGridView();
            gbAll.Controls.Add(dgvAllPlayers);

            // Права колонка: Розділена навпіл (Зверху - склад, 
знизу - турнірна таблиця балів)
            TableLayoutPanel rightLayout = new TableLayoutPanel
            {
                Dock = DockStyle.Fill,
                RowCount = 2,
                ColumnCount = 1
            };
            rightLayout.RowStyles.Add(new 
RowStyle(SizeType.Percent, 50.0f));
            rightLayout.RowStyles.Add(new 
RowStyle(SizeType.Percent, 50.0f));

            GroupBox gbPart = CreateStyledGroupBox("Склад 
поточного турніру");
            dgvParticipants = CreateStyledDataGridView();
            gbPart.Controls.Add(dgvParticipants);
```

```
            GroupBox gbLeaderboard = CreateStyledGroupBox("🏆 Поточна турнірна таблиця (Бали)");
            dgvLeaderboard = CreateStyledDataGridView();
            gbLeaderboard.Controls.Add(dgvLeaderboard);

            rightLayout.Controls.Add(gbPart, 0, 0);
            rightLayout.Controls.Add(gbLeaderboard, 0, 1);

            // 3. Панель кнопок (Центральна колонка)
            FlowLayoutPanel btnPanel = new FlowLayoutPanel
            {
                Dock = DockStyle.Fill,
                FlowDirection = FlowDirection.TopDown,
                WrapContents = false,
                Padding = new Padding(5, 40, 5, 0)
            };

            Size btnSize = new Size(155, 45);
            btnAddToTournament = CreateStyledButton("Додати ➡", Color.FromArgb(39, 174, 96), btnSize);
            btnRemoveFromTournament = CreateStyledButton("⬅ Прибрати", Color.FromArgb(192, 57, 43), btnSize);
            btnGenerateMatches = CreateStyledButton("🎲 Сформувати тур", Color.FromArgb(41, 128, 185), btnSize);
            btnShowMatches = CreateStyledButton("📋 Список ігор", Color.FromArgb(52, 73, 94), btnSize);
            btnRefreshLeaderboard = CreateStyledButton("🔄 Оновити бали", Color.FromArgb(142, 68, 173), btnSize); // Фіолетова кнопка

            // ПРИВ'ЯЗКА ПОДІЙ КЛІКУ ДЛЯ ВСІХ КНОПОК
            btnAddToTournament.Click += BtnAddToTournament_Click;
            btnRemoveFromTournament.Click += BtnRemoveFromTournament_Click;
            btnGenerateMatches.Click += BtnGenerateMatches_Click;

            btnRefreshLeaderboard.Click += (s, e) => {
                LoadLeaderboard(); // Оновлення лідерборду вручну
            };

            btnShowMatches.Click += (s, e) => {
                MatchesListForm mlf = new MatchesListForm(tournamentId);
                mlf.ShowDialog();
                LoadLeaderboard(); // Перерахунок балів автоматично після закриття форми матчів
            };

            btnPanel.Controls.AddRange(new Control[] {
                btnAddToTournament, btnRemoveFromTournament,
                new Label { Height = 30 },
```

```
                btnGenerateMatches,                 btnShowMatches,
btnRefreshLeaderboard
            });

            mainLayout.Controls.Add(gbAll, 0, 0);
            mainLayout.Controls.Add(btnPanel, 1, 0);
            mainLayout.Controls.Add(rightLayout, 2, 0);

            this.Controls.Add(mainLayout);
            this.Controls.Add(lblHeader);
        }

        private void LoadData()
        {
            try
            {
                using      (SqlConnection      conn      =      new
SqlConnection(connectionString))
                {
                    conn.Open();

                    // Гравці, яких ще немає в турнірі
                    string sqlAll = "SELECT PlayerID, FullName as
[ПІБ], CurrentRating as [Рейтинг] FROM Players " +
                                    "WHERE PlayerID NOT IN (SELECT
PlayerID FROM TournamentParticipants WHERE TournamentID = @tid)";
                    SqlDataAdapter        da1        =        new
SqlDataAdapter(sqlAll, conn);

da1.SelectCommand.Parameters.AddWithValue("@tid", tournamentId);
                    DataTable dt1 = new DataTable();
                    da1.Fill(dt1);
                    dgvAllPlayers.DataSource = dt1;

                    // Учасники турніру
                    string   sqlPart   =   "SELECT   p.PlayerID,
p.FullName as [ПІБ], p.Rank as [Розряд] FROM Players p " +
                                     "JOIN TournamentParticipants tp
ON p.PlayerID = tp.PlayerID WHERE tp.TournamentID = @tid";
                    SqlDataAdapter        da2        =        new
SqlDataAdapter(sqlPart, conn);

da2.SelectCommand.Parameters.AddWithValue("@tid", tournamentId);
                    DataTable dt2 = new DataTable();
                    da2.Fill(dt2);
                    dgvParticipants.DataSource = dt2;

                    if (dgvAllPlayers.Columns["PlayerID"] != null)
dgvAllPlayers.Columns["PlayerID"].Visible = false;
                    if   (dgvParticipants.Columns["PlayerID"]   !=
null) dgvParticipants.Columns["PlayerID"].Visible = false;
                }
            }
```

```
            catch (Exception ex) { MessageBox.Show("Помилка
завантаження даних: " + ex.Message); }
        }

        // Метод динамічного підрахунку очок та відображення
лідерборду
        private void LoadLeaderboard()
        {
            string query = @"
                SELECT
                    ROW_NUMBER() OVER (ORDER BY SUM(
                        CASE
                            WHEN m.WhitePlayerID = p.PlayerID AND
m.Result = 1 THEN 1.0
                            WHEN m.BlackPlayerID = p.PlayerID AND
m.Result = 0 THEN 1.0
                            WHEN (m.WhitePlayerID = p.PlayerID OR
m.BlackPlayerID = p.PlayerID) AND m.Result = 0.5 THEN 0.5
                            ELSE 0.0
                        END
                    ) DESC) AS [Місце],
                    p.FullName AS [Гравець],
                    p.Rank AS [Розряд],
                    SUM(
                        CASE
                            WHEN m.WhitePlayerID = p.PlayerID AND
m.Result = 1 THEN 1.0
                            WHEN m.BlackPlayerID = p.PlayerID AND
m.Result = 0 THEN 1.0
                            WHEN (m.WhitePlayerID = p.PlayerID OR
m.BlackPlayerID = p.PlayerID) AND m.Result = 0.5 THEN 0.5
                            ELSE 0.0
                        END
                    ) AS [Очки]
                FROM Players p
                JOIN TournamentParticipants tp ON p.PlayerID =
tp.PlayerID
                LEFT JOIN Matches m ON (p.PlayerID =
m.WhitePlayerID OR p.PlayerID = m.BlackPlayerID) AND
m.TournamentID = tp.TournamentID
                WHERE tp.TournamentID = @TournamentID
                GROUP BY p.PlayerID, p.FullName, p.Rank
                ORDER BY [Очки] DESC;";

            using (SqlConnection conn = new
SqlConnection(connectionString))
            {
                try
                {
                    conn.Open();
                    using (SqlCommand cmd = new SqlCommand(query,
conn))
                    {
```

```
cmd.Parameters.AddWithValue("@TournamentID", tournamentId);

                        SqlDataAdapter adapter = new
SqlDataAdapter(cmd);
                        DataTable dt = new DataTable();
                        adapter.Fill(dt);

                        dgvLeaderboard.DataSource = dt;
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Помилка оновлення таблиці
результатів: " + ex.Message, "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                }
            }
        }

        private void BtnGenerateMatches_Click(object sender,
EventArgs e)
        {
            string systemType = "";

            // 1. Отримуємо тип системи турніру
            using (SqlConnection conn = new
SqlConnection(connectionString))
            {
                conn.Open();
                SqlCommand cmd = new SqlCommand("SELECT SystemType
FROM Tournaments WHERE TournamentID = @tid", conn);
                cmd.Parameters.AddWithValue("@tid", tournamentId);
                systemType = cmd.ExecuteScalar()?.ToString() ??
"Колова";
            }

            List<int> playersToPair = new List<int>();

            try
            {
                using (SqlConnection conn = new
SqlConnection(connectionString))
                {
                    conn.Open();

                    // 2. Визначаємо номер наступного туру
                    SqlCommand cmdRound = new SqlCommand("SELECT
ISNULL(MAX(RoundNumber), 0) FROM Matches WHERE TournamentID =
@tid", conn);
                    cmdRound.Parameters.AddWithValue("@tid",
tournamentId);
```

```
                    int currentMaxRound = Convert.ToInt32(cmdRound.ExecuteScalar());
                    int nextRoundNumber = currentMaxRound + 1;

                    // ПЕРЕВІРКА: Шукаємо незіграні ігри поточного туру
                    if (currentMaxRound > 0)
                    {
                        SqlCommand cmdCheckUnfinished = new SqlCommand(
                            "SELECT COUNT(*) FROM Matches WHERE TournamentID = @tid AND RoundNumber = @round AND Result IS NULL", conn);
                        cmdCheckUnfinished.Parameters.AddWithValue("@tid", tournamentId);
                        cmdCheckUnfinished.Parameters.AddWithValue("@round", currentMaxRound);
                        int unfinishedCount = Convert.ToInt32(cmdCheckUnfinished.ExecuteScalar());

                        if (unfinishedCount > 0)
                        {
                            MessageBox.Show($"Неможливо створити тур №{nextRoundNumber}. У турі №{currentMaxRound} ще залишилися незіграні матчі або не внесено результати!",
                                "Попередження", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                            return;
                        }
                    }

                    // 3. Відбір гравців залежно від системи турніру та номеру туру
                    if (nextRoundNumber == 1)
                    {
                        foreach (DataGridViewRow row in dgvParticipants.Rows)
                        {
                            if (row.Cells["PlayerID"].Value != null)
                                playersToPair.Add(Convert.ToInt32(row.Cells["PlayerID"].Value));
                        }
                    }
                    else
                    {
                        if (systemType.Contains("виліт") || systemType.ToUpper().Contains("KNOCKOUT"))
                        {
                            // СИСТЕМА НА ВИЛІТ: Доопрацьований SQL-запит (переможці попереднього туру)
                            string sqlWinners = @"
```

```
                                SELECT DISTINCT
                                    CASE
                                        WHEN Result = 1 THEN
WhitePlayerID
                                        WHEN Result = 0 THEN
BlackPlayerID
                                        ELSE NULL
                                    END
                                FROM Matches
                                WHERE TournamentID = @tid AND
RoundNumber = @prevRound AND Result IS NOT NULL";

                            using (SqlCommand cmdWinners = new
SqlCommand(sqlWinners, conn))
                            {

cmdWinners.Parameters.AddWithValue("@tid", tournamentId);

cmdWinners.Parameters.AddWithValue("@prevRound", currentMaxRound);
                                using (SqlDataReader reader =
cmdWinners.ExecuteReader())
                                {
                                    while (reader.Read())
                                    {
                                        if (reader[0] !=
DBNull.Value && reader[0] != null)

playersToPair.Add(Convert.ToInt32(reader[0]));
                                    }
                                }
                            }
                        }
                        else
                        {
                            // КОЛОВА СИСТЕМА: Грають знову всі
учасники без виключення
                            foreach (DataGridViewRow row in
dgvParticipants.Rows)
                            {
                                if (row.Cells["PlayerID"].Value !=
null)

playersToPair.Add(Convert.ToInt32(row.Cells["PlayerID"].Value));
                            }
                        }
                    }

                    // 4. Перевірки на кількість учасників
                    if (playersToPair.Count < 2)
                    {
                        MessageBox.Show("Недостатньо гравців для
формування наступного туру. Турнір завершено або залишився один
```

```
фіналіст!",      "Турнір      завершено",      MessageBoxButtons.OK,
MessageBoxIcon.Information);
                        return;
                    }

                    //   5.   Алгоритм   жеребкування   (Випадкове
перемішування гравців)
                    Random rng = new Random();
                    var   shuffled   =   playersToPair.OrderBy(a   =>
rng.Next()).ToList();
                    int pairsCount = shuffled.Count / 2;

                    // 6. Запис нових ігор в базу даних
                    string   sqlInsert   =   "INSERT   INTO   Matches
(TournamentID, WhitePlayerID, BlackPlayerID, RoundNumber, Result,
IsPlayed, MatchDate) " +
                                           "VALUES    (@tid,    @white,
@black, @round, NULL, 0, GETDATE())";

                    int matchesCreated = 0;
                    for (int i = 0; i < pairsCount * 2; i += 2)
                    {
                        using    (SqlCommand    cmdInsert    =    new
SqlCommand(sqlInsert, conn))
                        {

cmdInsert.Parameters.AddWithValue("@tid", tournamentId);

cmdInsert.Parameters.AddWithValue("@white", shuffled[i]);

cmdInsert.Parameters.AddWithValue("@black", shuffled[i + 1]);

cmdInsert.Parameters.AddWithValue("@round", nextRoundNumber);
                            cmdInsert.ExecuteNonQuery();
                            matchesCreated++;
                        }
                    }

                    MessageBox.Show($"Жеребкування        проведено
успішно!\nСистема:           {systemType}\nСформовано           Тур
№{nextRoundNumber}\nСтворено матчів: {matchesCreated}",
                        "Успіх",              MessageBoxButtons.OK,
MessageBoxIcon.Information);

                    LoadLeaderboard();  //  Автоматично  оновлюємо
бали відразу після генерації туру
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка під час генерації туру: "
+     ex.Message,     "Помилка     БД",     MessageBoxButtons.OK,
MessageBoxIcon.Error);
```

```
            }
        }

        // --- Допоміжні методи для стилізації (Дизайн інтерфейсу)
---
        private GroupBox CreateStyledGroupBox(string title) => new
GroupBox
        {
            Text = title,
            Dock = DockStyle.Fill,
            Font = new Font("Segoe UI", 10, FontStyle.Bold),
            Padding = new Padding(10)
        };

        private  DataGridView  CreateStyledDataGridView()  =>  new
DataGridView
        {
            Dock = DockStyle.Fill,
            SelectionMode                                          =
DataGridViewSelectionMode.FullRowSelect,
            AllowUserToAddRows = false,
            ReadOnly = true,
            RowHeadersVisible = false,
            BackgroundColor = Color.White,
            BorderStyle = BorderStyle.None,
            AutoSizeColumnsMode                                    =
DataGridViewAutoSizeColumnsMode.Fill,
            EnableHeadersVisualStyles = false,
            ColumnHeadersHeight = 35
        };

        private  Button  CreateStyledButton(string  text,  Color
color, Size size) => new Button
        {
            Text = text,
            Size = size,
            BackColor = color,
            ForeColor = Color.White,
            FlatStyle = FlatStyle.Flat,
            Font = new Font("Segoe UI", 9, FontStyle.Bold),
            Margin = new Padding(0, 10, 0, 10),
            Cursor = Cursors.Hand
        };

        private  void  BtnAddToTournament_Click(object  sender,
EventArgs e)
        {
            if (dgvAllPlayers.SelectedRows.Count > 0)
            {
                int                        pid                     =
(int)dgvAllPlayers.SelectedRows[0].Cells["PlayerID"].Value;
                ExecuteAction("INSERT  INTO  TournamentParticipants
(TournamentID, PlayerID) VALUES (@tid, @pid)", pid);
```

```
                LoadLeaderboard();   //   Перераховуємо   бали,   бо
додався новий гравець
            }
        }

        private  void  BtnRemoveFromTournament_Click(object  sender,
EventArgs e)
        {
            if (dgvParticipants.SelectedRows.Count > 0)
            {
                int                          pid                          =
(int)dgvParticipants.SelectedRows[0].Cells["PlayerID"].Value;
                ExecuteAction("DELETE  FROM  TournamentParticipants
WHERE TournamentID = @tid AND PlayerID = @pid", pid);
                LoadLeaderboard();   //   Перераховуємо   бали,   бо
гравця видалено
            }
        }

        private void ExecuteAction(string sql, int pid)
        {
            try
            {
                using      (SqlConnection      conn      =      new
SqlConnection(connectionString))
                {
                    conn.Open();
                    SqlCommand cmd = new SqlCommand(sql, conn);
                    cmd.Parameters.AddWithValue("@tid",
tournamentId);
                    cmd.Parameters.AddWithValue("@pid", pid);
                    cmd.ExecuteNonQuery();
                }
                LoadData();
            }
            catch (Exception ex) { MessageBox.Show(ex.Message); }
        }
         }
                                                                    }
```

Додаток Б

SQL-скрипти створення таблиць реляційної бази даних інформаційної системи

```
CREATE TABLE [dbo].[Matches](
    [MatchID] [int] IDENTITY(1,1) NOT NULL,
    [TournamentID] [int] NULL,
    [RoundNumber] [int] NOT NULL,
    [WhitePlayerID] [int] NULL,
    [BlackPlayerID] [int] NULL,
    [Result] [float] NULL,
    [IsPlayed] [bit] NULL DEFAULT ((0)),
    [MatchDate] [datetime] NULL DEFAULT (GETDATE()),
    PRIMARY KEY CLUSTERED ([MatchID] ASC)
);
GO

CREATE TABLE [dbo].[Players](
    [PlayerID] [int] IDENTITY(1,1) NOT NULL,
    [FullName] [nvarchar](100) NOT NULL,
    [BirthDate] [date] NULL,
    [CurrentRating] [int] NULL DEFAULT ((1200)),
    [Rank] [nvarchar](50) NULL,
    [Club] [nvarchar](100) NULL,
    PRIMARY KEY CLUSTERED ([PlayerID] ASC)
);
GO

CREATE TABLE [dbo].[TournamentParticipants](
    [ParticipationID] [int] IDENTITY(1,1) NOT NULL,
    [TournamentID] [int] NULL,
    [PlayerID] [int] NULL,
    [TournamentPoints] [float] NULL DEFAULT ((0)),
    PRIMARY KEY CLUSTERED ([ParticipationID] ASC)
);
GO

CREATE TABLE [dbo].[Tournaments](
    [TournamentID] [int] IDENTITY(1,1) NOT NULL,
    [TournamentName] [nvarchar](200) NOT NULL,
    [SystemType] [nvarchar](50) NULL,
    [StartDate] [date] NULL,
    [EndDate] [date] NULL,
    [TimeControl] [nvarchar](50) NULL,
    [IsActive] [bit] NULL DEFAULT ((1)),
    PRIMARY KEY CLUSTERED ([TournamentID] ASC)
);
GO

ALTER TABLE [dbo].[Matches] WITH CHECK ADD FOREIGN
KEY([BlackPlayerID])
```

```
REFERENCES [dbo].[Players] ([PlayerID]);
GO

ALTER TABLE [dbo].[Matches] WITH CHECK ADD FOREIGN
KEY([TournamentID])
REFERENCES [dbo].[Tournaments] ([TournamentID]);
GO

ALTER TABLE [dbo].[Matches] WITH CHECK ADD FOREIGN
KEY([WhitePlayerID])
REFERENCES [dbo].[Players] ([PlayerID]);
GO

ALTER TABLE [dbo].[TournamentParticipants] WITH CHECK ADD FOREIGN
KEY([PlayerID])
REFERENCES [dbo].[Players] ([PlayerID]) ON DELETE CASCADE;
GO

ALTER TABLE [dbo].[TournamentParticipants] WITH CHECK ADD FOREIGN
KEY([TournamentID])
REFERENCES [dbo].[Tournaments] ([TournamentID]) ON DELETE CASCADE;
GO
```