

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ФОРМУВАННЯ СКЛАДУ ФУТБОЛЬНОЇ КОМАНДИ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR
FORMING A FOOTBALL TEAM

Виконав: студент(ка) 2 курсу, групи КНТ-129м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Зайцев Д.А.

(прізвище та ініціали)

Керівник Терещенко Е.В.

(прізвище та ініціали)

Рецензент Морщавка С.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2020 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Зайцева Дениса Андрійовича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів
формування складу футбольної команди
Research and Software Implementation of Methods for Forming a Football Team

керівник проєкту (роботи) Терещенко Еліна Валентинівна, к.ф.-м.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “3” листопада 2020 року № 301

2. Строк подання студентом проєкту (роботи) грудень 2020 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження.
2. Матеріали і методи. 3. Опис програми. 4. Розробка програми. 5. Експлуатація
та тестування програми. 6. Економіко-організаційна частина, 7. Охорона праці
та безпека в надзвичайних ситуаціях.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Терещенко Е.В., доцент		
Економіко-організаційна частина	Пожуєва Т.О., професор		
Охорона праці та безпека в надзвичайних ситуаціях	Коробко О.В., ст. викладач		
Нормоконтролер	Дейнега Л.Ю., ст. викладач		

7. Дата видачі завдання “ 7 ” вересня 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Розробка методу	3–4 тижні	Розділ 2
4	Розробка архітектури програми	5 тиждень	Розділ 3
5	Розробка програми	6–7 тижні	Розділ 4
6	Тестування та відлагодження програмного забезпечення	8 тиждень	Розділ 5
7	Розробка організаційно-економічної частини	9 тиждень	Розділ 6
8	Розробка заходів з охорони праці та цивільної безпеки	10 тиждень	Розділ 7
9	Оформлення пояснювальної записки та документів до неї	11 тиждень	Додатки
10	Нормоконтроль та рецензування	12 тиждень	
11	Захист роботи	13 тиждень	

Студент(ка)

(підпис) Зайцев Д.А.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Терещенко Е.В.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
132 с., 28 рис., 9 табл., 3 дод., 45 джерел.

ФУТБОЛ, ФОРМУВАННЯ СКЛАДУ КОМАНДИ, ФУТБОЛЬНА
СТАТИСТИКА, РЕКУРЕНТНІ НЕЙРОННІ МЕРЕЖІ, МОВА
ПРОГРАМУВАННЯ C#, MYSQL.

Об'єкт дослідження – накопичення та оброблення футбольної статистики. Предмет дослідження – програмне забезпечення формування складу футбольної команди.

Мета роботи – програмна реалізація системи підтримки прийняття рішень щодо формування складу футбольної команди на основі накопиченої статистики футбольних змагань.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мова програмування C#, система MySQL, рекурентні нейронні мережі.

Результати. Розроблено програмне забезпечення, яке дозволяє накопичувати й переглядати дані футбольної статистики, що включають результати виступу в окремих іграх, накопичені дані за турніром та кар'єрою гравця, виконувати прогнозування виступу гравця на турнірі та автоматизовано формувати оптимальний склад футбольної команди.

Висновки. Запропоновано метод формування складу футбольної команди на основі оптимізації сумарної оцінки виступу гравців команди. На основі запропонованого методу розроблено програмне забезпечення за результатами аналізу предметної області, проєктування, реалізації та тестування програми.

Галузь використання – футбольний менеджмент, футбольна статистика, віртуальні футбольні турніри.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 132 pages, 28 figures, 9 tables, 3 appendixes, 45 sources.

FOOTBALL, FOOTBALL TEAM FORMING, FOOTBALL STATISTICS, RECURRENT NEURAL NETWORKS, C# PROGRAMMING LANGUAGE, MYSQL.

The object of research is the accumulation and processing of football statistics. The subject of research is software for forming a football team.

The purpose of the work is in the program implementation of the decision support system for the formation of the football team on the basis of the accumulated statistics of football competitions.

Materials, methods and technical means: object-oriented programming, C# programming language, MySQL system, recurrent neural networks.

Results. The software has been developed. It allows to collect and view football statistics, including the results of performances in individual games, the accumulated data on the tournament and career of players, to predict the performance of players in the tournament and automatically form the optimal football team.

Conclusions. The method of forming a football team based on the optimization of the total assessment of the players performance is proposed. Based on the proposed method, software is developed based on the results of subject area analysis, design, implementation and testing of the program.

Fields of usage include football management, football statistics, virtual football tournaments.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	8
Вступ	9
1 Аналіз проблеми та постановка завдань дослідження	10
1.1 Формування складу футбольної команди за правилами «фентезі-футбол»	10
1.2 Огляд програмних аналогів	14
1.3 Висновки за розділом 1 та постановка завдань дослідження	20
2 Матеріали і методи	22
2.1 Постановка задачі формування складу футбольної команди	22
2.2 Метод динамічного програмування	24
2.3 Обґрунтування вибору моделей на основі нейронних мереж для прогнозування кількості набраних очок гравцем	26
2.4 Вибір моделі на основі нейронних мереж для прогнозування кількості набраних очок гравцем	27
2.5 Метод формування складу футбольної команди	29
2.6 Висновки за розділом 2	31
3 Опис програми	33
3.1 Визначення вимог до програмного забезпечення	33
3.2 Вибір мови програмування	36
3.3 Проектування бази даних	37
3.4 Висновки за розділом 3	43
4 Розробка програми	44
4.1 Реалізація структури програми	44
4.2 Реалізація класів програми	46
4.3 Висновки за розділом 4	48
5 Експлуатація та тестування програми	49
5.1 Призначення та умови роботи з програмою	49
5.2 Опис роботи з програмою	49

5.3 Тестування запропонованого методу формування складу футбольної команди.....	56
6 Економіко-організаційна частина	58
6.1 Концепція економічного обґрунтування.....	58
6.2 Планування розробки програмного продукту	58
6.3 Розрахунок основної заробітної плати	61
6.4 Розрахунок додаткової заробітної плати.....	61
6.5 Єдиний соціальний внесок.....	61
6.6 Визначення витрат на матеріали	62
6.7 Витрати на спеціальне обладнання	63
6.8 Розрахунок накладних витрат.....	66
6.9 Розрахунок економічної ефективності програмного продукту	67
7 Охорона праці та безпека в надзвичайних ситуаціях.....	70
7.1 Аналіз потенційних небезпек	70
7.2 Заходи по забезпеченню безпеки.....	70
7.3 Заходи по забезпеченню виробничої санітарії та гігієни праці	71
7.4 Заходи по забезпеченню пожежної безпеки	77
7.5 Заходи по забезпеченню безпеки у надзвичайних ситуаціях.....	78
Висновки.....	80
Перелік джерел посилання	81
Додаток А Технічне завдання.....	86
Додаток Б Текст програми.....	90
Додаток В Слайди презентації	124

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- ВХУ – вогнища хімічного ураження;
ЕОМ – електронно-обчислювальна машина;
ОЦ – обчислювальний центр;
ПК – персональний комп'ютер;
ПУЕ – правила улаштування електроустановок.

ВСТУП

Футбол часто називають «спорт номер один в світі» – цей вид спорту популярний у будь-якій точці на планеті і має повне право називатися найпопулярнішим спортом в світі, але також футбольна індустрія не стоїть на місці і, продовжуючи розвиватися семимильними кроками, змушує звертати на себе увагу багатьох значущих соціальних інститутів [1].

Операційні доходи футбольних клубів на даний момент складають величезні суми. Так найбільший операційний дохід за результатами 2019 року однієї команди складає 839,5 мільйонів євро [2]. Такі показники є непідйомними для багатьох інших цілих галузей, не говорячи про окремі підприємства в таких галузях.

Розвиток футболу, привертання уваги величезної кількості людей, а разом з тим і отримані від цього кошти призвели до того, що однією з його активних частин стали інформаційні технології, які відповідно є одним з локомотивів розвитку футболу зараз. Так чи інакше, будь-яка команда користується інформаційними технологіями. Менеджери все частіше користуються програмними засобами для прийняття власних рішень. Менеджери команд найвищого рівня використовують їх для аналізу суперників, аналізу власної команди, відповідно для цього потрібно накопичувати певну футбольну статистику. Одним з найважливіших завдань менеджера є завдання вибору складу футбольної команди. Саме цьому завданню і присвячено дану роботу. Завдання розглядається з одного боку в розрізі формування складу футбольної команди, а з іншого боку – в розрізі формування складу віртуальної футбольної команди для участі в кіберспортивних змаганнях на основі реальних виступів футболістів.

Метою роботи є розроблення програмного забезпечення формування складу футбольної команди на основі накопиченої статистики футбольних змагань для забезпечення автоматизованого прийняття рішень.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Формування складу футбольної команди за правилами «фентезі-футбол»

Фентезі-футбол – це гра, де користувач виконує роль власника, генерального менеджера і тренера команди, обирає реальних гравців, що беруть участь у одній з ліг світового футболу, приносять очки, в залежності від свого виступу в реальній лізі, очки обраховуються загалом наступним чином: плюсується кількість очок, зароблених за голи, гольові передачі, виходи на поле тощо, а віднімається кількість очок за пропущені командою голи, отримані картки тощо [3].

Щодо правил, які використовуються під час формування такої команди, можна виділити наступне:

- команда збирається на певний бюджет (зазвичай це 100 балів);
- у сформованій збірної повинні бути, як і в реальному футболі, воротар, захисник, нападник, півзахисник – всього близько 11 гравців (11 або більше, якщо дозволяються запасні);
- можна зібрати команду з різних футбольних клубів або ж з одного, але частіше за все всі гравці вибираються з одного дивізіону певної країни;
- чим більше очок потенційно може заробити гравець, тим вище його ціна, тому через обмежений бюджет не вдасться зібрати всіх кращих гравців в одну команду;
- коли остаточний склад сформований, учасник його зберігає, в подальшому в кожному турі буде можливість провести три трансфери (обмін вже обраних гравців на нових), необхідність таких трансферів викликана травмами обраних гравців або ж участю обраних гравців в матчах, де їх команда не є фаворитом;
- після складання збірної, в реальному житті, гравці беруть участь в спортивних змаганнях і приносять бали, тому завдання – сформувати якомога більш вдалий колектив, тобто підібрати гравців так, щоб вони

зробили під час гри максимально прибуткові досягнення: наприклад, якщо член команди визнаний гравцем матчу, забив гол або є воротарем і відбив пенальті, то все це приносить певну кількість балів у скарбничку, але варто пам'ятати і про негативні пункти (якщо член сформованої збірної отримав жовту або червону картку);

- виходячи з набраних очок, учасник турніру розташовується в турнірній таблиці;

- тривалість турніру зазвичай збігається з тривалістю сезону в чемпіонаті по футболу тієї чи іншої країни [4]-[5].

На практиці прогнозування кількості набраних очок виконується, враховуючи такі фактори:

- схеми і тактика: наприклад, в останні роки деякі команди грають в три центральні захисники, що призвело до появи ряду гравців, які числяться захисниками, але насправді є вінгерами, при цьому в цих гравців вище потенціал голів і передач, досвідчені менеджери виявлять таких гравців і користуються даною логікою, а ті, хто виявить це після першого ігрового тижня, будуть змушені використовувати додаткові трансфери, втрачати на цьому очки, гравці поза позиції можуть нести як позитивний, так і негативний потенціал (наприклад, деякі гравці числяться півзахисниками, хоча загалом є нападниками, і набирають за кожен гол на очко більше, ніж нападники, і отримують ще очко, якщо команда завершує гру без пропущених голів, з іншого боку, номінальний вінгер може грати в захисті, а це означає, що такий гравець набирає менше очок за відсутність пропущених голів і забиті голи);

- пенальті і стандарти: у багатьох командах є штатні пенальтисти, але виникають ситуації, коли тренер змінює власну позицію, штатний пенальтист отримує травму або переходить в інший клуб, тому гравець, який виконує стандарти, може бути досить цінним активом і вигідним диференціалом, слід також звернути увагу на тих, хто виконує штрафні удари, з яких можна забити (навіть якщо вони не привели до голу), це може стати джерелом

несподіваних очок протягом сезону, точно так само і з кутовими (їх іноді виконують захисники, які можуть заробити гольові передачі);

– очікувані голи і очікувані гольові передачі: вимірюють, скільки голів і передач гравець повинен був зробити, враховуючи можливості, які у нього були в грі, кожному удару або гольовий передачі гравця присвоюється значення (засноване на середньому результаті удару або спроби гольової передачі з тієї ж позиції на полі), чим більше значення очікуваних голів і передач, тим більша ймовірність того, що гравець забив або асистував, якщо дельта між таким показником і фактично забитими м'ячами велика, можна очікувати регресу до середнього, відповідно у протилежному випадку можна очікувати, що рано чи пізно гравець скористається шансами і, можливо, варто його взяти в команду;

– важливий критерій – схильність гравця до травм: травми – це частина футболу, але дуже важко передбачити, коли гравець отримає травму, однак деякі гравці більш схильні до травм, ніж інші;

– дисципліна: гравці, схильні до порушень, не тільки отримують негативні очки за жовту і червону картки, але і піддаються дискваліфікації, картки під час передсезонної підготовки гравці отримують не часто, але, якщо це відбувається, потрібно задуматися, чи дійсно потрібен цей гравець;

– небезпечні наслідки напруженого літа: гравці, які взяли участь в останніх стадіях чемпіонату світу, Європи, Америки, швидше за все, отримають відпустку, почнуть грати в деяких чемпіонатах не з першого туру, і навпаки команди, тренери яких рано викликають команду на передсезонну підготовку, до старту сезону будуть підготовлені краще (хоча пізніше протягом сезону такі команди можуть бути схильні до втоми і ротації);

– важливими психологічним фактором є питання відсутності суперечок у гравця за контрактом: якщо гравець веде переговори про підписання нового контракту, він постарается показати хороші результати, щоб отримати максимальний зиск зі своїх переговорів;

– існує чітка тенденція зі збільшення частини бюджету команди, яка використовується на оборону, це відбувається через все більше поширення атакуючих захисників і використання високого пресингу, схема «4-4-2» вийшла з загального використання, а разом з нею і використання традиційних вінгера, ті, хто колись вважався вінгером, тепер грає як форвард в передній трійці, а захисники беруть на себе відповідальність за забезпечення ширини атаки, граючи на позиції крайніх захисників;

– високі витрати на півзахист збільшують ймовірність гарного виступу команди, за статистикою в основному ті, хто витрачали на свій півзахист більше, опинялися вище ніж ті, хто фінішував нижче, при цьому деяке зменшення суми витрат на півзахисників відбувалось за рахунок розподілу більших коштів на преміальних захисників [6].

Менеджер у грі «фентезі-футбол» відповідає не тільки за початкове формування складу команди, але і за підбір складу кожного тижня для кожного наступного туру чемпіонату. Окрім того менеджер відповідальний за трансфери гравців, якщо їх потрібно здійснити протягом тижня.

Під час вибору команди на тиждень враховується наступне:

– 11 гравців основного складу команди FPL повинні в чемпіонаті бути також основними гравцями: тренери роблять заміни близько 60-ї хвилини, потрібно обирати гравців, які грають всі 90 хвилин;

– одна з найпоширеніших помилок менеджерів полягає в тому, що вони набирають команду, орієнтуючись лише на хороші матчі, загальне емпіричне правило полягає в тому, щоб дивитися щонайменше на три ігрові тижні вперед і використовувати тільки тих гравців, які мають сприятливі матчі в цей період;

– дивитися потрібно на результати далі трьох тижнів, більш того, потрібно вивчати не загальну складність суперників, а окремо наступальну і оборонну складність [6].

Таким чином «фентезі-футбол» є одним з можливих варіантів оцінювання гравців футбольних команд, на основі чого можна отримувати

дані про реальні виступи гравців, а також формувати склад футбольної команди.

1.2 Огляд програмних аналогів

На даний момент не існує повного аналогу програмного забезпечення, що має розроблятися в цій роботі. Загалом таке рішення знаходиться на стику двох напрямків – застосунків, що реалізують турніри «фентезі футбол», хоча в даному випадку має використовуватися саме система обчислення очок, а з іншого боку – застосунків, що реалізують доступ до даних футбольної статистики.

Однією з найвідоміших реалізацій футбольних турнірів «фентезі-футбол» є gaming.uefa.com, що є реалізацією від Співки європейських футбольних асоціацій.

По-перше, під час аналізу даного застосунку було проаналізовано систему визначення очок, набраних гравцем, що використовується в даному застосунку.

В табличному вигляді система приведена в табл. 1.1.

Таблиця 1.1 – Система визначення очок, набраних гравцем

Категорія	Дія	Кількість очок
1	2	3
Всі гравці		
	Поява на полі	1
	60 хвилин, проведених на полі	2
	Гол з дальньої відстані (з-за меж штрафного майданчика)	1
	Результативна передача	3
	Кожні 3 відбори	1

Продовження таблиці 1.1

1	2	3
	Зароблення пенальті для команди	2
	Зароблення пенальті у власні ворота	-1
	Незабитий пенальті	-2
	Жовта картка	-1
	Червона картка	-3
	Автогол	-2
Воротар		
	Гол	6
	Відбитий пенальті	5
	Матч без пропущених голів (відіграно не менше 60 хвилин)	4
	Кожні три порятунки	1
	Кожні два пропущені голи	-1
Захисник		
	Гол	6
	Матч без пропущених голів (відіграно не менше 60 хвилин)	4
	Кожні два пропущені голи	-1
Півзахисник		
	Гол	5
	Матч без пропущених голів (відіграно не менше 60 хвилин)	1
Нападник		
	Гол	4

На рис. 1.1 приведено інтерфейс формування складу футбольної команди застосунку gaming.uefa.com [7].

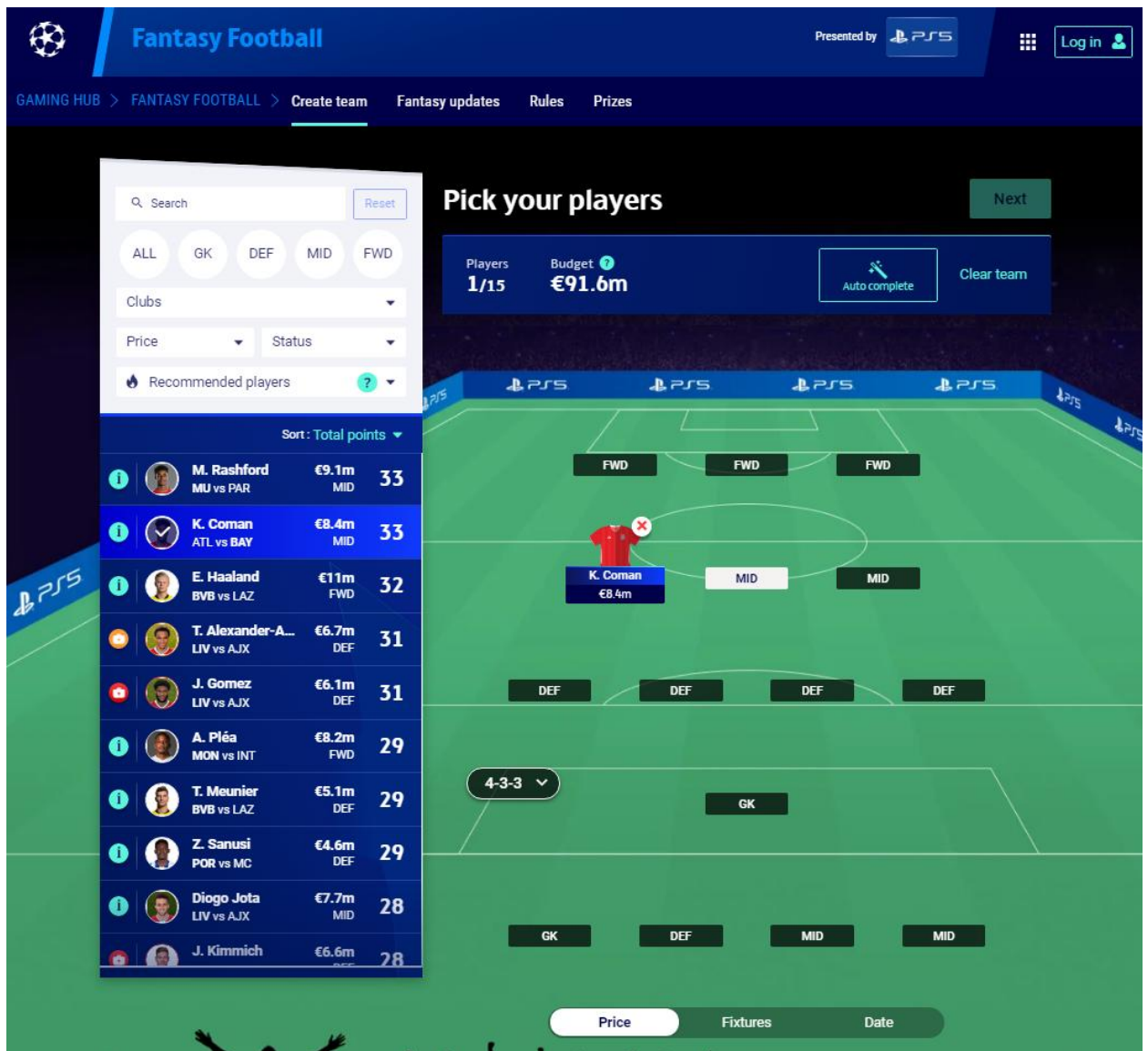


Рисунок 1.1 – Інтерфейс застосунку gaming.uefa.com

Правила Спілки європейських футбольних асоціацій вимагають початкового вибору складу з 15 футболістів, з яких кожного тижня будуть обиратися 11 гравців основного складу, при цьому 15 футболістів обов'язково включають:

- два воротаря;
- п'ять захисників;
- п'ять захисників;
- три нападники.

Формуючи основний склад на тиждень, потрібно обрати капітана. Очки, які він набрав протягом ігрового тижня, подвоюються.

Після закінчення туру виконуються автоматичні заміни, можна виставити своїх запасних в потрібному порядку на той випадок, якщо хтось із основних в звітному турі не зіграє:

- якщо воротар не вийшов на поле, він буде замінений на запасного воротаря за умови, що той взяв участь в минулих матчах;

- якщо польовий гравець не вийшов на поле, він буде замінений на запасного польового: буде обрано запасного в зазначеному порядку за умови, що він взяв участь в минулих матчах і його заміна не порушує правила формування складу;

- в основному складі завжди повинні бути один воротар, як мінімум три захисника, два півзахисника і один нападник, тобто, якщо в тактичній розстановці три захисника, але один з них не зіграв, він може бути замінений тільки на одного з двох запасних захисників [8].

В останні роки було додано такі правила до гри:

- введено так звані бустери: за допомогою бустерів можна зробити необмежену кількість змін у своїй команді, всього бустерів два, і кожен з них можна використовувати всього один раз;

- введено систему джокерів: джокер дозволяє зробити необмежену кількість безкоштовних трансферів перед найближчим туром, всі куплені гравці залишаються в команді по завершенні туру, правила тренерських заміні, автоматичних заміні, зміни капітана, формування складу та ліміту на гравців з однієї команди залишаються в силі;

- якщо вже було зроблено заміни перед туром, а потім вирішено використовувати джокера, то всі попередні трансфери будуть скасовані, як і відрахування очок за них, якщо було перевищено число безкоштовних трансферів;

- при використанні джокера на груповому етапі один безкоштовний трансфер переноситься на наступний тур, якщо Джокер використовується в плейофф, додаткові безкоштовні трансфери не нараховуються;

– використовувати джокер можна один раз за турнір, після активації його не можна скасувати;

– безлімітний: безлімітний бустер дозволяє змінити команду тільки на один тур, він дозволяє створити команду без обмежень по бюджету і кількості трансферів, по завершенні туру буде той же склад, який був до використання безлімітного бустера,

– якщо заміни зроблено перед туром, а потім вирішено використовувати безлімітний бустер, то всі попередні трансфери будуть скасовані, як і відрахування очок за них, якщо перевищено число безкоштовних трансферів;

– при використанні безлімітного бустера на груповому етапі один безкоштовний трансфер переноситься на наступний тур, якщо бустер використовується в плейофф, додаткові безкоштовні трансфери не нараховуються;

– використовувати безлімітний бустер можна один раз за турнір. Після активації його не можна скасувати [8].

Існує багато варіацій «фентезі-футбол», в яких дещо змінено правила, в яких надаються можливості гри проти суперника, створення окремих ліг і т.д.

На рис. 1.2 приведено приклад інтерфейсу програмного застосунку fanteam.com [9].

Організація інтерфейсу [fanteam](http://fanteam.com) достатньо схожа, хоча має і засоби певного прогнозування дій гравців на тур. Зокрема під час вибору гравця можна побачити, чи очікується вихід гравця в основному складі, чи це менш імовірна подія.

Особливістю [fanteam](http://fanteam.com) є те, що в даному випадку організовуються одноденні турніри, тобто склад обирається на один раз: один день або тур, а не на весь турнір.

FanTeam's Festive Freerolls: Day 1 - UCL
UEFA Champions League

Buy in: €0.00 | Entries: 788 | 03:50:34

Details | Scoring | Prizepool

ENTER TOURNAMENT (FREE)

Players selected: 2 / 6

Set as captain | Set as vice captain

Fill the rest

PURSUIT ? Bonus points: -4.32

GOALKEEPER

DEFENDER

MIDFIELDER

FORWARD

Status information

- Player is Confirmed for lineup
- Player is Expected for lineup
- Player is Confirmed for bench
- Player Might start
- Player is Not expected to start or Confirmed not to play
- Player is Injured or Suspended

If two or more teams are tied in a Pursuit game, the team with the highest amount of bonus points will rank the lowest. If they are still tied, the team that joined first, will win the prize

Рисунок 1.2 – Інтерфейс застосунку fanteam.com

Вебсайти футбольної статистики є основою для вибору складу команди, адже якщо вибір складу відбувається вручну, то потрібно розуміти результати виступу гравців за минулі періоди. Для цього і мають використовуватися такі вебсайти. Найвідомішим вебсайтом даного типу є whoscored.com [10] (рис. 1.3).

У даному випадку можна переглянути таблиці турнірів, результати окремих ігор, а також статистику за кожною грою, що включає зокрема статистику виступу кожного окремого гравця в грі і включає багато різних показників (голи, удари по воротах, кількість передач, кількість точних передач, зони володіння м'ячем тощо).



Рисунок 1.3 – Інтерфейс застосунку whoscored.com

1.3 Висновки за розділом 1 та постановка завдань дослідження

Проаналізовано правила формування складу футбольної команди за принципом «фентезі-футбол». Розглянуто принципи, за якими менеджери обирають склад команд. Розглянуто систему оцінювання гравців на основі очок за виконані ними дії на футбольному полі. Проаналізовано програмні аналоги, що включають gaming.uefa.com, fanteam.com, whoscored.com.

В якості завдань дослідження виділено наступні:

- розроблення методу формування складу футбольної команди;
- проєктування програмного забезпечення;
- програмна реалізація та тестування програмного забезпечення.

2 МАТЕРІАЛИ І МЕТОДИ

Загалом система очок, набраних гравцями в фентезі-футбол – це один зі способів, як можна оцінити виступ гравців у турнірі. Тобто система, яка використовується для оцінювання гравців у грі, може бути основою утворення рейтингу гравців, адже оцінювання виконується на основі реальних дій гравців, а не віртуальних.

Формування складу футбольної команди на основі оцінки як в реальній грі, так і в фентезі-футбол може бути простою задачею, допоки не розглядати всі можливі варіанти рішень, що приймаються. У такому випадку вибір оптимального рішення є достатньо складною задачею, що потребує математичної формалізації і програмного розв'язання.

2.1 Постановка задачі формування складу футбольної команди

Під час вибору складу футбольної команди потрібно зі всієї множини наявних гравців на футбольному ринку обрати таких, які дозволять принести сумарно найбільшу можливу кількість очок. Маючи історичні дані щодо виступу гравців у минулому, потрібно виконати прогнозування виступу гравців у майбутньому. Саме прогнозовані очки, які наберуть гравці в майбутньому, тобто під час виступу у турнірі, є основою прийняття рішень. Команда має бути сформована з 11 гравців (або іншої заданої кількості n) таким чином, щоб обраний склад гравців забезпечив максимальну сумарну прогнозовану кількість очок, яку набере команда.

При цьому команда не може дозволити придбати собі будь-якого гравця, адже має обмежену суму коштів. Ця сума коштів – величина P . Сумарна вартість 11 гравців має бути не більшою за значення P .

Тобто задача формування складу футбольної команди є задачею максимізації з введеними обмеженнями.

Одночасно потрібно враховувати, що не може бути обрано склад команди з довільних гравців, тому що команда повинна грати за певною схемою, що визначає тренер перед реалізацією такого вибору. Тобто в команді не може бути 11 воротарів або 11 захисників.

Схема визначає кількість воротарів n_g , кількість захисників n_d , кількість півзахисників n_m та кількість нападників n_f . Сумарно кількість цих чотирьох складових дорівнює n .

Якщо кількість гравців дорівнює 11, то $n_g = 1$.

У такому випадку задача формування складу футбольної команди може бути виражена у вигляді задачі оптимізації (2.1)-(2.8):

$$\sum_{j=1}^n s_j(a_{j-1}, d_j) \rightarrow \max, \quad (2.1)$$

$$n = n_g + n_d + n_m + n_f, \quad (2.2)$$

$$P_g + P_d + P_m + P_f \leq P, \quad (2.3)$$

$$P_g = \sum_{j=1}^{n_g} p_{g_j}, \quad (2.4)$$

$$P_d = \sum_{j=1}^{n_d} p_{d_j}, \quad (2.5)$$

$$P_m = \sum_{j=1}^{n_m} p_{m_j}, \quad (2.6)$$

$$P_f = \sum_{j=1}^{n_f} p_{f_j}, \quad (2.7)$$

$$d_j \in F, \quad (2.8)$$

де a_{j-1} – стан команди, якого було досягнуто на попередньому, $j - 1$, кроці;

d_j – рішення про вибір гравця в команду, що приймається на j -му кроці і виконується з множини F – множини всіх гравців, доступних на ринку;

$s_j(a_{j-1}, d_j)$ – сумарна оцінювана (прогнозована) кількість очок, яку набере команда, сформована на j -му кроці;

n – кількість гравців у команді;

n_g – кількість воротарів у команді;

n_d – кількість захисників у команді;

n_m – кількість півзахисників у команді;
 n_f – кількість нападників у команді;
 P_g – сумарна вартість воротарів, за яку їх було придбано в команду;
 P_d – сумарна вартість захисників, за яку їх було придбано в команду;
 P_m – сумарна вартість півзахисників, за яку їх було придбано в команду;
 P_f – сумарна вартість нападників, за яку їх було придбано в команду;
 P – максимальний бюджет команди, тобто сумарна вартість всіх гравців команди;
 p_{g_j} – вартість воротаря з номером g_j з інтервалу від 1 до n_g ;
 p_{d_j} – вартість воротаря з номером g_j з інтервалу від 1 до n_d ;
 p_{m_j} – вартість воротаря з номером g_j з інтервалу від 1 до n_m ;
 p_{f_j} – вартість воротаря з номером g_j з інтервалу від 1 до n_f .

2.2 Метод динамічного програмування

Парадигма «розділяй і володарюй» – важлива парадигма розробки алгоритмів, яка полягає в рекурсивному розбитті розв'язуваної задачі на дві або більше підзадачі того ж типу, але меншого розміру, і комбінуванні їх рішень для отримання відповіді до вихідної задачі, розбиття виконуються до тих пір, поки всі підзадачі не виявляться елементарними [11].

Динамічне програмування – спосіб вирішення складних завдань шляхом розбиття їх на більш прості підзадачі: динамічне програмування, як і парадигма «розділяй і володарюй», дозволяє вирішувати завдання, комбінуючи рішення допоміжних завдань, відмінність цього методу в тому, що допоміжні завдання не є незалежними, тобто різні допоміжні завдання використовують рішення одних і тих же підзадач:

а) динамічне програмування є методом, який підходить до вирішення деякого класу задач шляхом їх розкладання на частини, невеликі та менш складні завдання, при цьому характерною рисою є рішення задач по етапах,

через фіксовані інтервали, проміжки часу, що і визначило появу терміна динамічне програмування, слід зауважити, що методи динамічного програмування успішно застосовуються і при вирішенні завдань, в яких фактор часу не враховується, в цілому метод можна представити як покрокове або поетапне програмування, рішення задач методами динамічного програмування проводиться на основі сформульованого принципу оптимальності Р. Е. Беллманом: оптимальна поведінка має ту властивість, що яким би не був первісний стан системи і початкове рішення, подальше рішення має визначати оптимальну поведінку відносно стану, отриманого в результаті початкового рішення;

б) принцип оптимальності Беллмана: якщо деяка послідовність рішень оптимальна, то на будь-якому етапі наступні рішення утворюють оптимальну стратегію по відношенню до результату попередніх рішень, тобто яким би не був стан системи перед черговим кроком, треба вибрати управління на цьому кроці так, щоб виграш на даному кроці (програш) плюс оптимальний виграш (програш) на всіх наступних кроках був би максимальним (мінімальним), на основі принципу оптимальності Беллмана будується схема рішення багатокрокового завдання, що складається з 2-х частин: зворотний хід (від останнього кроку до першого отримують множину можливих оптимальних («умовно-оптимальних») управлінь), прямий хід (від відомого початкового стану до останнього з отриманої множини «умовно-оптимальних» управлінь складається шукане оптимальне управління для всього процесу в цілому);

в) динамічне програмування корисно, якщо на різних шляхах багаторазово зустрічаються одні і ті ж підзадачі, основний технічний прийом – запам'ятовувати рішення підзадач, що зустрічаються, на випадок, якщо та ж підзадача зустрінеється знову;

г) у типовому випадку динамічне програмування застосовується до завдань оптимізації, у такого завдання може бути багато можливих рішень, але потрібно вибрати оптимальне рішення, при якому значення деякого параметра буде мінімальним або максимальним;

д) типовий алгоритм вирішення задач методом динамічного програмування:

- 1) описати будову оптимальних рішень;
- 2) виписати рекурентне співвідношення, що зв'язують оптимальні значення параметра для підзадач;
- 3) рухаючись від низу до верху, обчислити оптимальне значення параметра для підзадач;
- 4) користуючись отриманою інформацією, побудувати оптимальне рішення [11]-[13].

2.3 Обґрунтування вибору моделей на основі нейронних мереж для прогнозування кількості набраних очок гравцем

Окрім розв'язання безпосередньо задачі оптимізації (2.1)-(2.8) необхідно обрати механізм, який буде використовуватися для прогнозування кількості набраних очок гравцем, адже цей показник є підґрунтям для оптимізації. Використання нейронних мереж в задачах оптимізації зокрема досліджувалось в роботах [14]-[15].

Задача прогнозування полягає у визначенні майбутнього або майбутніх станів системи за її минулими станами. У випадку прогнозування кількості набраних очок гравцем минулими станами є дані про кількість очок, набрану гравцем за попередні сезони, а майбутнім станом – кількість очок, яку гравець набере у наступному сезоні. Задачі даного типу є задачами прогнозування часового ряду. Для їх розв'язання на даний момент активно використовуються нейронні мережі. Зокрема даним питанням стосовно різних прикладних галузей присвячено роботи [16]-[20].

Стандартним механізмом, який використовується для прогнозування є лінійна регресія, що за своєю суттю дозволяє усереднити минулий досвід, та інші традиційні статистичні методи.

Порівнюючи ці два підходи можна виділити наступне:

– на відміну від лінійної регресії нейронні мережі направлені не тільки на відтворення досвіду, який був отриманий в галузі за минулий період, а на прогнозування майбутнього стану на основі минулого досвіду;

– нейронна мережа не вимагає відсіювати окремо неінформативні екземпляри даних, а отримує інформацію про такі екземпляри в процесі навчання, тобто даний процес виявляється автоматизованим;

– нейронна мережа має більшу адаптивність, що дозволяє працювати в умовах зміни середовища, що важливо зокрема, наприклад, зараз в умовах пандемії;

– нейронна мережа дещо прискорює час самого прийняття рішень, адже після навчання мережі відома її повна структура, а обчислення виходу за заданими входами займає менше часу ніж обчислення залежностей за всіма даними, особливо в деяких нелінійних статистичних методах.

Тому в результаті було прийнято рішення щодо вибору нейронних мереж в якості моделі прогнозування кількості набраних очок гравцем.

2.4 Вибір моделі на основі нейронних мереж для прогнозування кількості набраних очок гравцем

За результатами аналізу робіт [16]-[20] було виділено, що двома основними архітектурами, які використовуються для прогнозування є повнозв'язні нейронні мережі та рекурентні нейронні мережі. Результати порівняння даних двох архітектур було зведено до табл. 2.1.

Оскільки рекурентні нейронні мережі дозволяють визначати взаємозалежності, тобто залежності між вхідними даними, а не тільки залежності між входом і виходом, що важливо для розв'язання даної задачі, оскільки виступ гравця в одному сезоні має вплив на його виступ в наступному сезоні, то в результаті було обрано саме рекурентні нейронні мережі для прогнозування кількості набраних очок гравцем.

На даний момент можна виділити наступні архітектури рекурентних нейронних мереж:

- прості рекурентні нейронні мережі;
- long short term memory;
- gated recurrent units;
- двонаправлені рекурентні нейронні мережі;
- квазірекурентні нейронні мережі [21].

Таблиця 2.1 – Порівняння архітектур нейронних мереж для розв’язання задач прогнозування

Критерій	Повнозв’язні нейронні мережі	Рекурентні нейронні мережі
Можливість прогнозування часового ряду	+	+
Врахування взаємозалежностей між значеннями	–	+
Область застосування	Універсальна	Прогнозування фінансових рядів, аналіз і генерація тексту

При виборі архітектури нейронної мережі необхідно брати до уваги, що зазвичай футболіст може активно брати участь у великих турнірах до 15 разів, а частіше ця цифра менше. Отже, для задач прогнозування часового ряду кількість сезонів є достатньо невеликою, тому для реалізації прогнозування було застосовано прості рекурентні нейронні мережі, які є дещо спрощеним варіантом відносно long short term memory.

2.5 Метод формування складу футбольної команди

У даному підрозділі необхідно узагальнити рішення, які було прийнято в попередніх підрозділах, і сформулювати метод формування складу футбольної команди.

Відповідно для розв'язання задачі (2.1)-(2.8) потрібно обрати оптимальний варіант зі всіх можливих рішень, кожне з яких складається з n кроків, на кожному з яких виконується вибір з множини всіх гравців F .

Тоді задачу вибору можна перетворити на задачу вибору оптимального розподілення коштів між сумами P_g, P_d, P_m, P_f . При цьому зазвичай в іграх фентезі-футбол вибір виконується для $P = 100$, а всі суми p_{g_j} і так далі для кожної лінії гравців задаються зі значенням до десятої.

У такому випадку в множині всіх можливих розв'язань задачі можна виділити повторення оптимальної підзадачі щодо вибору оптимального складу для кожної лінії за заданої суми витрат на цю лінію. Такі оптимальні підзадачі постійно повторюються. А для розв'язання підзадачі потрібно відсортувати гравців у заданій лінії (воротарі, захисники, півзахисники, нападаючі) за очікуваною кількістю очок, а далі обрати n_g перших позицій для воротарів (аналогічно для захисників, півзахисників, нападаючих), які сумарно коштують менше P_g .

Побудуємо метод формування складу футбольної команди, який складається з п'яти етапів.

На першому етапі методу має бути зібрано набір даних, що характеризують виступ кожного гравця, що буде брати участь у турнірі, на попередніх турнірах. З цих даних має бути сформована вибірка даних, що складається з k елементів на вході та 1 елемента на виході. При чому якщо екземпляр початкової вибірки складається з більшої кількості елементів за k , то екземпляр розбивається у вибірці на декілька методом ковзаючого вікна, тобто k перших, $k+1$ на виході, а потім зі зсувом на 1 наступний екземпляр і так далі.

Фактично під час розв’язання практичної задачі, описаної в розділі 5, було використано розбиття набору даних на 5 вхідних значень і одне вихідне, тобто на основі кількості очок, які було набрано гравцем в 5 сезонах, що відбулись поспіль, реалізується прогнозування очок, набраних гравцем в шостому сезоні.

На другому етапі виконується створення моделі нейронної мережі. Модель створюється на основі рекурентної нейронної мережі, яку було обрано в підрозділі 2.5. За результатами експериментів на практиці було використано просту рекурентну нейронну мережу, що складається з наступних шарів:

- вхідного шару, який складається з 5 нейронів, що відповідають кількості очок, набраних гравцем за 5 сезонів поспіль;
- прихованого рекурентного шару, який містить 200 нейронів з функцією активації ReLU;
- прихованого рекурентного шару, який містить 100 нейронів з функцією активації ReLU;
- вихідного повнозв’язного шару з 1 нейроном на виході, що визначає кількість очок, яку набере гравець в наступному (шостому сезоні).

На третьому етапі реалізується навчання рекурентної нейронної мережі по сформованій на першому етапі вибірці. Для оцінювання результатів навчання використовувалась середньо-квадратична помилка, а в якості оптимізатора – алгоритм Адама.

У випадку простої рекурентної нейронної мережі обчислення виходу одного блоку, з яких складається вся мережа, виконується за допомогою формули (2.9):

$$S_k = f(S_{k-1} \cdot W_{rec} + X_k \cdot W_x + b), \quad (2.9)$$

де S_k – стан в момент часу k ;

b – поріг;

S_{k-1} – попередній стан в момент часу $k - 1$;

X_k – вхід в момент часу k ;

W_{rec}, W_x – вагові параметри;

f – функція [22].

На четвертому етапі забезпечується прогнозування виступу кожного гравця за k попередніми значеннями для $k+1$ невідомого періоду часу. Фактично відбувалося прогнозування результату шостого сезону за значеннями попередніх п'яти.

На п'ятому етапі розв'язується задача оптимізації (2.1)-(2.8) методом динамічного програмування.

Оптимальний вибір є результатом роботи методу формування складу футбольної команди і рекомендується менеджеру футбольної команди для прийняття рішення.

Під час застосування на практиці етапи 1-3 можуть використовуватися один раз, а за необхідності через деякий час (наприклад, при отриманні великої кількості нових актуальних даних) повторюватися, а етапи 4-5 реалізуються циклічно, оскільки для кожного наступного сезону, а також в процесі реалізації сезону потрібно прогнозувати за заданих вхідних даних про гравця результати його майбутнього виступу.

2.6 Висновки за розділом 2

Виконано математичну постановку задачі формування складу футбольної команди.

Проаналізовано метод динамічного програмування, який запропоновано для використання в розв'язанні сформульованої задачі оптимізації. Запропоновано використовувати систему очок, розглянуту в розділі 1, для оцінювання гравців.

Запропоновано використовувати рекурентні нейронні мережі для прогнозування майбутніх результатів виступу гравців. Запропоновано метод формування складу футбольної команди.

Наукова новизна роботи полягає в запропонованому методі формування складу футбольної команди на основі оптимізації сумарної оцінки виступу футбольної команди, виконуючи оптимізацію на основі методу динамічного програмування, а прогнозування індивідуальних виступів на основі рекурентних нейронних мереж, що дозволяє в підсумку виконувати автоматизоване прийняття рішень щодо вибору складу футбольної команди з обмеженням щодо вартості.

3 ОПИС ПРОГРАМИ

3.1 Визначення вимог до програмного забезпечення

За результатами проведеного огляду в розділі 1 було вирішено розробляти систему підтримки прийняття рішень щодо формування складу футбольної команди як застосунок з доступом до функцій накопичення та доступу до футбольних даних (футбольної статистики). Застосунок має бути розроблено для настільної операційної системи сімейства Windows, але при цьому можливе з часом розширення платформи зі створенням програмних засобів для інших платформ. Тобто обов'язковим є створення бази даних для накопичення даних, а також механізму автоматизованого формування складу футбольної команди.

До функцій, які має реалізовувати програмне забезпечення, було віднесено:

- внесення даних про футбольні турніри, що включає можливості створення нових турнірів, редагування існуючих та за необхідності видалення існуючих;

- внесення даних про футбольні команди, що включає можливості внесення даних про нові команди, редагування існуючих та за необхідності видалення існуючих;

- внесення даних про гравців футбольних команд, зокрема додавання нових гравців, редагування даних про існуючих та за необхідності видалення існуючих;

- додавання футбольних матчів, що проводяться у межах відповідного турніру;

- внесення результатів футбольних матчів, включаючи редагування даних на різних етапах;

- внесення статистичних показників футбольних матчів, зокрема відсотку володіння м'ячем, точності передач команди, кількості нанесених ударів по воротах;

- внесення результатів виступу гравців у футбольних матчах;
- керування набором котирувань;
- внесення даних про фактичні котирування відповідних подій;
- обчислення статистики гравців і результатів команд на основі результатів окремих матчів або турнірів загалом;
 - вибір складу власної футбольної команди з можливістю автоматичного формування складу команди;
 - прогнозування результатів виступу гравців;
 - перегляд найкращих футбольних гравців за обраними користувачем критеріями;
 - перегляд найкращих футбольних команд за обраними користувачем критеріями;
 - обчислення статистики виступу обраної користувачем футбольної команди;
 - відображення результатів проведення футбольного турніру, в тому числі поточних результатів;
 - відображення даних гравця загалом та в межах обраного футбольного матчу;
 - відображення даних команди загалом та в межах обраного футбольного матчу;
 - внесення ставок на події;
 - відображення виконаних ставок;
 - автоматизоване підведення підсумків за виконаними користувачами ставками;
 - авторизація і реєстрація облікових записів.

Для визначення сценаріїв взаємодії користувачів та адміністраторів з програмою була створена діаграма прецедентів (рис. 3.1).

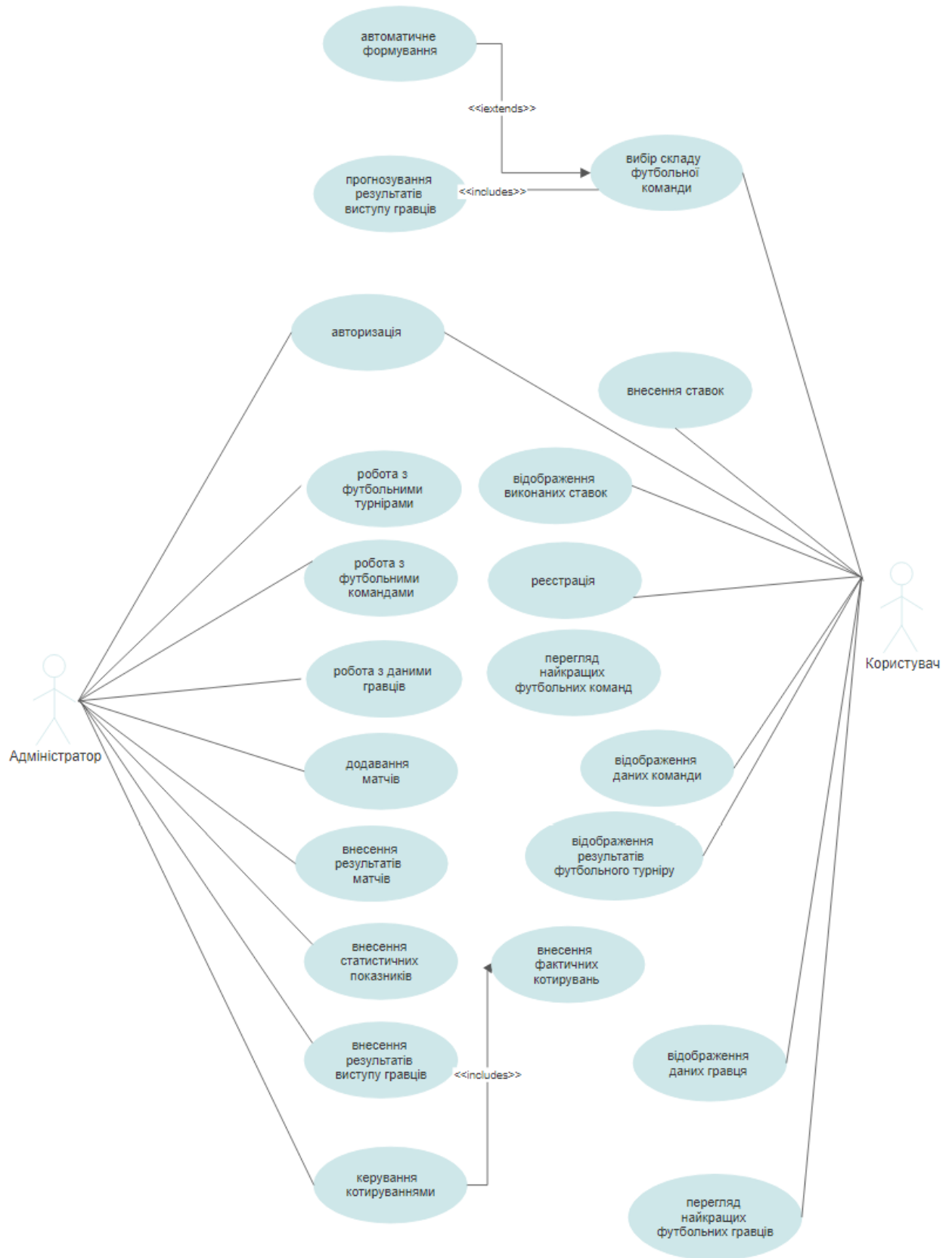


Рисунок 3.1 – Діаграма прецедентів

3.2 Вибір мови програмування

Зважаючи, що застосунок є настільним, основною областю використання буде Windows, то для реалізації було розглянуто мови C# та C++.

Підсумки порівняння було зведено до табл. 3.1.

Таблиця 3.1 – Порівняння мов програмування

Критерій	C#	C++
Парадигми програмування	Мультипарадигмальна мова з об'єктно-орієнтованим програмуванням на основі низькорівневих абстракцій	Мультипарадигмальна мова з об'єктно-орієнтованим програмуванням на основі високорівневих абстракцій
Залежність від платформи	Від платформи не залежить	Від платформи не залежить, але основний напрямок – Windows
Підтримка типів даних	Статична, обмежена	Статична/динамічна, більша кількість
Область застосування	Застосунки для настільних систем, вебзастосунки	Ігри, драйвери для пристроїв, серверні застосунки

У підсумку обрано мову C# завдяки вищому рівню абстракції, зручнішому набору типів та системі їх побудови, а також тому, що настільні застосунки є прямою її областю застосування.

3.3 Проектування бази даних

Базу даних було створено для збереження даних футбольної статистики та участі віртуальної команди в турнірі.

Для реалізації системи управління базою даних було використано MySQL, оскільки вона має наступні позитивні властивості:

- безкоштовна;
- кросплатформна, що дозволяє запускати її на сервері без особливих вимог щодо його операційної системи;
- має високу продуктивність.

Перелік таблиць бази даних описано нижче.

У таблиці Player наводяться дані про гравця, включаючи такі характеристики:

- id – ідентифікаційний номер (первинний ключ, тип int);
- name – ім'я (тип varchar (50));
- nation – національність (тип varchar (50));
- team_id – команда (зовнішній ключ з таблиці Team, тип int);
- birthday – дата народження (тип date);
- price – поточна вартість гравця (тип real);
- games – кількість ігор, яку загалом провів гравець (тип int);
- goals – кількість голів, яку сумарно забив гравець (тип int);
- goalpasses – кількість результативних передач (тип int);
- attempts – сумарна кількість ударів гравця (тип int);
- goalattempts – сумарна кількість голевих моментів гравця (тип int);
- ycards – кількість жовтих карток, які сумарно отримав гравець (тип int);
- rcards – кількість червоних карток, які сумарно отримав гравець (тип int);
- picking – сумарна кількість відборів гравця (тип int);
- foals – сумарна кількість порушень правил гравця (тип int);

- passes – сумарна кількість передач гравця (тип int);
- height – висота гравця (тип int);
- weight – вага гравця (тип int);
- num – номер гравця (тип int);
- position – позиція на полі гравця (тип int);
- predscores – кількість очок, яку за прогнозом має отримати гравець за сезон (тип int).

У таблиці Team наводяться дані про команду, включаючи такі характеристики:

- id – ідентифікаційний номер (первинний ключ, тип int);
- name – назва (тип varchar (50));
- city – місто (тип varchar (30));
- stadium – назва стадіону (тип varchar (50)).

У таблиці Tournament наводяться дані про команду, включаючи такі характеристики:

- id – ідентифікаційний номер (первинний ключ, тип int);
- name – назва (тип varchar (50));
- num – кількість учасників (тип int);
- tstart – дата початку (тип date);
- tdate – дата завершення (тип date).

У таблиці TournamentTeam наводяться дані про виступ команди у турнірі, включаючи такі характеристики:

- id – ідентифікаційний номер (первинний ключ, тип int);
- tournament_id – номер турніру (зовнішній ключ з таблиці Tournament, тип int);
- team_id – номер команди (зовнішній ключ з таблиці Team, тип int);
- place – місце в таблиці турніру (тип int);
- score – набрані очки в турнірі (тип int);
- games – кількість проведених ігор в турнірі (тип int);
- wins – кількість перемог в турнірі (тип int);

- draws – кількість нічиїх в турнірі (тип int);
- losses – кількість поразок в турнірі (тип int);
- ogoals – кількість забитих м'ячів в турнірі (тип int);
- agoals – кількість пропущених м'ячів в турнірі (тип int).

У таблиці Game наводяться дані про гру, включаючи такі характеристики:

- id – ідентифікаційний номер (первинний ключ, тип int);
- tournament_id – номер турніру (зовнішній ключ з таблиці Tournament, тип int);
- num – номер гри за порядком в турнірі (тип int);
- gdate – дата проведення гри (тип date);
- goals1 – кількість голів, забитих першою командою (тип int);
- goals2 – кількість голів, забитих другою командою (тип int);
- team1_id – номер першої команди (зовнішній ключ з таблиці Team, тип int);
- team2_id – номер другої команди (зовнішній ключ з таблиці Team, тип int);
- type – тип запису (гра не розпочалась, гра активна, гра завершилась, тип bool).

У таблиці PlayerGame розташовані дані, що відповідають результатам участі гравця у грі, включаючи такі характеристики:

- id – ідентифікаційний номер (первинний ключ, тип int);
- player_id – номер гравця (зовнішній ключ з таблиці Player, тип int);
- game_id – номер гри (зовнішній ключ з таблиці Game, тип int);
- goals – кількість голів (тип int);
- longgoals – кількість голів з дальньої дистанції (тип int);
- goalpasses – кількість результативних передач (тип int);
- attempts – кількість ударів гравця (тип int);
- goalattempts – кількість голевих моментів гравця (тип int);
- ycards – кількість жовтих карток (тип int);

- rcards – кількість червоних карток (тип int);
- picking – кількість відборів гравця (тип int);
- foals – кількість порушень правил гравця (тип int);
- passes – кількість передач гравця (тип int);
- position – позиція на полі гравця у грі (тип int);
- minutes – кількість хвилин на полі (тип int);
- penalties – кількість забитих голів з пенальті (тип int);
- pensaved – кількість відбитих пенальті гравцем (тип int);
- penfor – кількість зароблених гравцем пенальті (тип int);
- penagainst – кількість пенальті у власні ворота, отриманих через цього гравця (тип int);
- penlost – кількість пропущених пенальті (тип int);
- owngoals – кількість автоголів (тип int);
- goalslost – кількість пропущених голів за час гри гравця (тип int);
- saves – кількість порятунків (тип int);
- scores – кількість очок, набраних гравцем за системою «фентезі-футбол» (тип int);
- predscores – кількість очок, яку за прогнозом має отримати гравець за цю гру (тип int).

У таблиці User розташовані дані, що відповідають користувачам, включаючи такі характеристики:

- id – ідентифікаційний номер (первинний ключ, тип int);
- login – логін (тип varchar (20));
- password – пароль (тип varchar (20));
- name – ім'я (тип varchar (50));
- type – тип запису (користувач, адміністратор, тип int).

У таблиці UserTeam наводяться дані про команду користувача, включаючи такі характеристики:

- id – ідентифікаційний номер (первинний ключ, тип int);

- user_id – номер користувача (зовнішній ключ з таблиці User, тип int);
- tournament_id – номер турніру (зовнішній ключ з таблиці Tournament, тип int);
- name – назва користувацької команди (тип varchar (100));
- score – кількість очок, зароблена користувацькою командою за турнір (тип int).

У таблиці UserTeamPlayer наводяться дані про команду користувача, включаючи такі характеристики:

- id – ідентифікаційний номер (первинний ключ, тип int);
- player_id – номер гравця (зовнішній ключ з таблиці Player, тип int);
- userteam_id – номер користувацької команди (зовнішній ключ з таблиці UserTeam, тип int);
- score – кількість очок, набраних гравцем (тип int).

У таблиці Event наводяться дані про подію, на яку виконуються ставки, включаючи такі характеристики:

- id – ідентифікаційний номер (первинний ключ, тип int);
- game_id – номер гри (зовнішній ключ з таблиці Game, тип int);
- info – опис події (тип varchar (100));
- coeff – розмір котирування (тип real);
- flag – реалізовано в результаті (так або ні, тип bool).

У таблиці Rate наводяться дані про ставку:

- id – ідентифікаційний номер (первинний ключ, тип int);
- user_id – номер користувача зовнішній ключ з таблиці User (тип int);
- event_id – подія, на яку виконано ставку (зовнішній ключ з таблиці Event, тип int);
- rtime – час, коли було зроблено ставку (тип datetime);
- rvalue – сума ставки (тип real);
- flag – реалізовано в результаті (так або ні, тип bool).

На рис. 3.2 представлено схему бази даних.

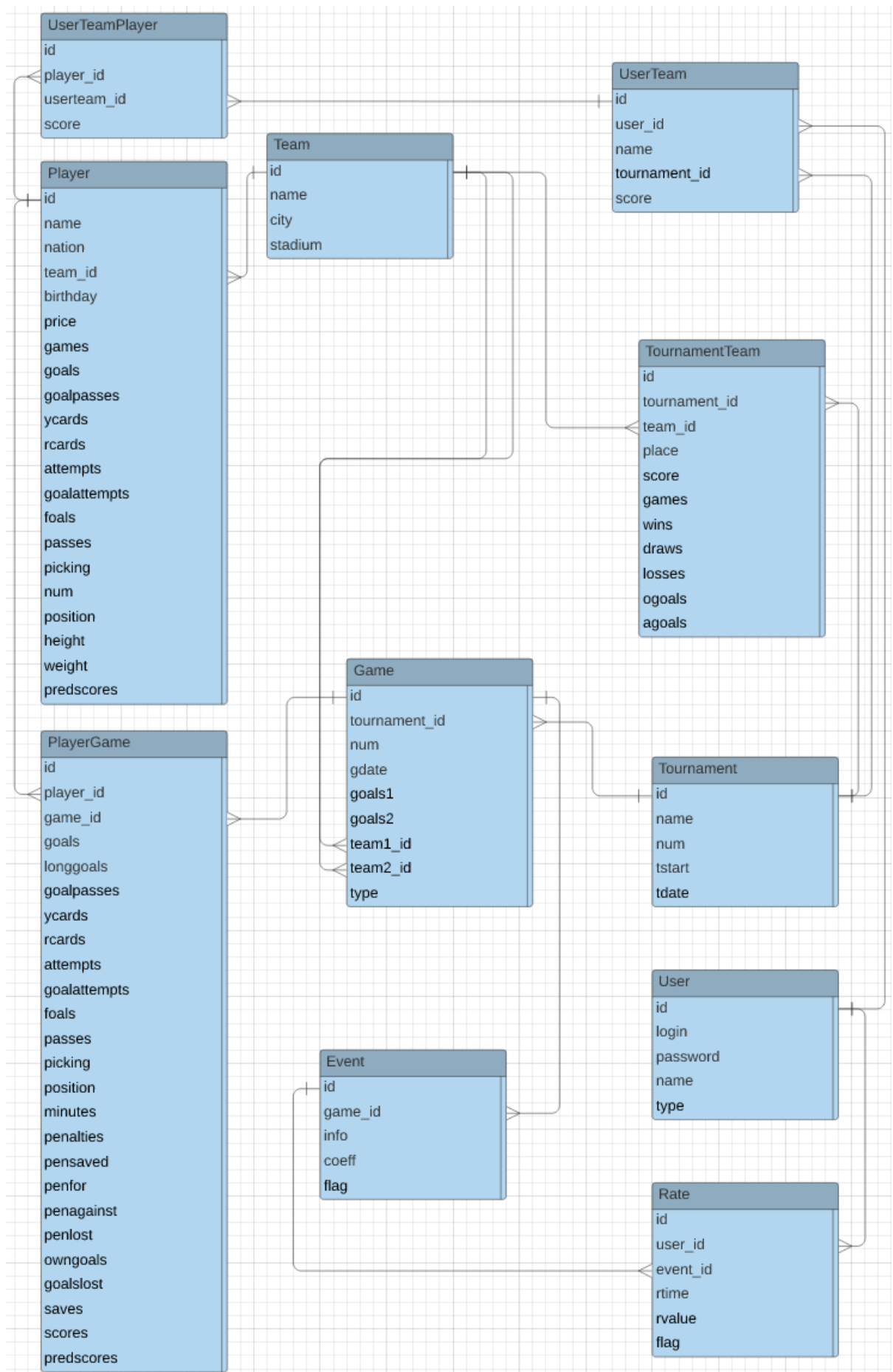


Рисунок 3.2 – Схема бази даних

3.4 Висновки за розділом 3

За підсумками виконання третього розділу було отримано наступні результати:

- вирішено виконувати розробку у вигляді настільного застосунку;
- визначено вимоги до програмного забезпечення;
- виділено основні сценарії роботи адміністратора та користувача з програмою;
- обрано мову C# для реалізації програмного забезпечення;
- реалізовано визначення структури бази даних за допомогою використання MySQL в якості інструменту роботи з бази даними.

4 РОЗРОБКА ПРОГРАМИ

4.1 Реалізація структури програми

Структура розробленої програми представлена на схемі на рис. 4.1.



Рисунок 4.1 – Схема структури програми

Програмне забезпечення складається з наступних одиниць:

- модуля з'єднання з базою даних на основі класу ConClass, що реалізує створення і керування з'єднанням, необхідним для реалізації функціональності в підсистемах;
- модуля авторизації на основі класу Authorize, який забезпечує виконання аутентифікації облікових записів;
- підсистеми графічного інтерфейсу, що на основі набору класів реалізує функціональність програми;
- підсистеми виконання оптимального вибору, що реалізує запропонований метод на основі класу оптимального вибору OptimForming і

класу SRNNModel, що реалізує модель на основі простої рекурентної нейронної мережі.

Створення класу SRNNModel реалізовано на основі застосування бібліотеки RNNSharp, що реалізує глибокі рекурентні нейронні мережі, однією з архітектур яких є проста рекурентна нейронна мережа.

До складу підсистеми графічного інтерфейсу зокрема входять наступні класи:

- FootballerStatisticsFrm – форма перегляду статистики виступу гравця;
- BestPlFrm – форма перегляду найращих гравців;
- MyTeamFrm – форма формування складу футбольної команди;
- TeamRatingFrm – форма загального рейтингу виступу сформованої команди серед усіх;
- TournamentsFrm – форма перегляду результатів турнірів;
- TeamsFrm – форма перегляду статистики команд;
- ProfileFrm – форма профілю облікового запису;
- AuthorizeFrm – форма авторизації облікового запису;
- RateFrm – форма створення власних ставок;
- RateResultsFrm – форма перегляду результатів власних ставок;
- RegisterFrm – форма реєстрації облікового запису;
- AddTournamentFrm – форма додавання і редагування даних турнірів;
- AddTeamFrm – форма додавання і редагування даних команд;
- AddFootballerFrm – форма додавання і редагування даних гравців;
- AddGameFrm – форма додавання даних матчів;
- AddRateFrm – форма додавання і редагування ставок;
- FixResultGameFrm – форма внесення результатів ігор;
- FixFootballerGameResultFrm – форма внесення статистики виступу гравців у іграх.

4.2 Реалізація класів програми

Реалізація класів виконана на основі взаємодії з базою даних. Основні класи представляють собою засоби реалізації функціональності на основі графічного інтерфейсу.

Відповідно для виконання базових дій застосовується клас ConClass, який реалізує зв'язування з базою даних. Даний клас має наступні методи.

Метод `void Create()` виконує з'єднання з базою даних і відкриття з'єднання, якщо воно в даний момент закрито:

```
if (cparams.State == System.Data.ConnectionState.Closed)
    cparams.Open();
```

Параметри з'єднання зберігаються в об'єкті класу `MySQLConnection`, представленому атрибутом `cparams`, де визначається адреса сервера, логін, пароль підключення, назва бази даних.

Метод `void Exit()` реалізує закриття з'єднання з базою даних, якщо воно було встановлено.

Метод `MySQLConnection getConn()` повертає встановлене на поточний момент з'єднання.

Під час реалізації графічної форми дані отримуються з бази даних на основі виконання відповідного SQL-запиту, отримані дані зберігаються в елементах графічного інтерфейсу користувача.

Приклад SQL-запиту для формування переліку доступних команд з обраного турніру з номером `seltournum` для забезпечення користувачем вибору з випадającego переліку:

```
conn.Create();

MySQLCommand msc = new MySQLCommand("SELECT Team.name " +
```

```

"FROM Team INNER JOIN TournamentTeam ON
TournamentTeam.team_id = Team.id " +
"WHERE TournamentTeam.tournament_id = @tid " +
"ORDER BY Team.name", conn.getConn());
msc.Parameters.Add("@tid", MySqlDbType.Int).Value = this.seltournum;

```

При цьому запит формується на основі класу `MySqlCommand`, а параметр визначається за допомогою форми `@tid` в тексті запиту, після чого до запиту додається значення цього параметра.

Далі на основі використання класу `MySqlDataReader` відбувається виконання сформованого SQL-запиту з його розбором та заповненням переліку:

```

MySqlDataReader msdr = msc.ExecuteReader();
if (msdr.HasRows)
{
    while (msdr.Read())
    {
        this.AllFTeamsCB.Items.Add(msdr["name"]);
        ForListElem fl;
        fl.id = MSDR["id"];
        fl.sd = MSDR["name"];
        teamlist.add(fl);
    }
}

```

Основні індекси даних, з якими відбувається робота на формі, зберігаються в атрибутах класа для спрощення подальшого доступу до них з бази даних за необхідності. Наприклад, в класі `FootballerStatisticsFrm` для

фіксації вибраних індексів сезону, турніру, команди, гравця та гри використовується відповідно набір атрибутів:

- int selseasonnum – обраний в інтерфейсі сезон футбольного турніру;
- List <ForListElem> seasonlist – перелік доступних сезонів;
- int selteamnum – індекс обраної в інтерфейсі футбольної команди;
- List <ForListElem> teamlist – перелік доступних футбольних команд;
- int seltournum – індекс обраного в інтерфейсі футбольного турніру;
- List <ForListElem> tournlist – перелік доступних футбольних турнірів;
- int selfootbnum – індекс обраного в інтерфейсі футболіста;
- List <ForListElem> footnlist – перелік доступних футболістів;
- int selgamenum – індекс обраної в інтерфейсі гри;
- List <ForListElem> gamelist – перелік доступних ігор.

Для цього також виділяються окремі структури. Наприклад, для формування елемента звіту щодо найкращих гравців в класі BestPlFrm виділено структуру ResultListElem, що включає наступні елементи:

- int id – номер гравця;
- string name – ім'я гравця;
- string team – назва команди;
- float fd – результат гравця (кількість забитих голів, результативних передач тощо).

4.3 Висновки за розділом 4

Описано розроблені модулі програми на основі схеми структури програми. Описано процес функціонування програми на основі схеми функціонування. Описано принципи реалізації основних класів програми та взаємодії з базою даних.

5 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

5.1 Призначення та умови роботи з програмою

Розроблене програмне забезпечення виконує з одного боку збір футбольної статистики за допомогою накопичення широкої статистики відносно кожного матчу кожного турніру, який відслідковується, а з іншого боку надає можливість виконувати формування власної команди на основі вибору вручну або автоматизованого прийняття відповідного рішення.

Мінімальні вимоги до складу та параметрів технічних засобів сервера для запуску системи:

- процесор: 4 ядра, частота – 2 ГГц;
- оперативний запам'ятовуючий пристрій: обсяг – 8 Гб;
- жорсткий диск: обсяг – 100 Гб.

5.2 Опис роботи з програмою

Для того щоб розпочати роботу з програмою, потрібно запустити на виконання файл FootballManager.exe, тоді на формі авторизації задати дані облікового запису користувача. У випадку визначення облікового запису адміністратора відкриється форма керування даними, з якої можна обрати потрібний розділ.

Так за допомогою кнопки «Турніри» можна перейти до форми керування турнірами (рис. 5.1). З цієї форми можна повернутися назад на форму керування даними загалом за допомогою кнопки «Назад» або вийти з облікового запису за допомогою кнопки «Вийти».

Надаються дві вкладки для роботи з турнірами: «Новий» і «Існуючий».

Вкладка «Новий» призначена для створення нового турніру. Необхідно ввести назву турніру, задати кількість команд, задати початкову дату та дату завершення турніру. За допомогою цих дат потім визначається,

до якого сезону відноситься турнір (шляхом виділення року початку і року завершення).

Турніри

Назад Вийти

Новий Існуючий

Назва Іспанська Ла Ліга

Кількість команд 20

Початок 01.09.2021

Завершення 15.05.2022

Створити

Сезон 2020/2021

Створені

- Англійська Прем'єр-ліга
- Іспанська Ла Ліга
- Німецька Бундесліга
- Італійська Серія А
- Французька Ліга 1

Рисунок 5.1 – Керування турнірами

У випадковому переліку «Сезон» можна обрати один з доступних сезонів, тоді в переліку нижче відобразиться набір всіх створених турнірів для обраного сезону.

Аналогічним чином реалізується вкладка «Існуючий», призначена для редагування та видалення існуючих (з переліку «Створені») турнірів.

Визначення ігор в межах турніру виконується за допомогою форми «Нова гра» (рис. 5.2).

Назад

Нова гра

Вийти

Сезон 2020/2021

Турнір Англійська Прем'єр-ліга

Тур 12

Перша команда Манчестер Юнайтед

Друга команда Манчестер Сіті

Дата

← грудень 2020 р. →

	Пн	Вт	Ср	Чт	Пт	Сб	Нд
49	30	1	2	3	4	5	6
50	7	8	9	10	11	12	13
51	14	15	16	17	18	19	20
52	21	22	23	24	25	26	27
53	28	29	30	31	1	2	3
2	4	5	6	7	8	9	10

 Сьогодні: 04.12.2020

Нова ставка

Коефіцієнт

Додати

Ставки

Перемога першої команди	2.1
Перемога другої команди	1.9
Нічия	1.4

Створити

Рисунок 5.2 – Додавання нової гри до календаря турніру

Форма «Нова гра» дозволяє визначити сезон та турнір, до якого належить гра, яку потрібно додати до календаря. Для цього потрібно з відповідних випадаючих переліків обрати сезон та турнір. Далі потрібно з переліку «Тур» обрати номер туру, до якого належить гра. Після цього з команд турніру потрібно обрати першу та другу команди. Далі потрібно з календаря обрати дату, коли відбувається гра.

Одночасно на цій формі можна визначити ставки для гри. Тут можна переглянути котирування, які вже було визначено в таблиці «Ставки», а можна за допомогою випадаючого переліку «Нова ставка» та текстового поля «Коефіцієнт» обрати потрібну ставку на матч і задати котирування, після чого за допомогою кнопки «Додати» створити запис про ставку.

На завершення потрібно за допомогою кнопки «Створити» зафіксувати матч в базі даних.

Якщо в програмі було авторизовано звичайного користувача, то для нього відкривається форма, побудована дещо інакше відносно форм адміністратора.

На рис. 5.3 наведено приклад форми «Інформація про гравців». Дана форма дозволяє переглядати дані щодо виступу гравців у турнірі як загалом, так і у конкретному матчі. Вкладка «Сумарне» дозволяє переглядати загальну статистику, а вкладка «Матч» дозволяє переглядати статистику за окремою грою, тобто на ній потрібно додатково до обраних параметрів ще обрати конкретну гру команди в даному турнірі.

Інформація про гравців

- [Моя команда](#)
- [Турніри](#)
- [Команди](#)
- [Найкращі](#)
- [Профіль](#)
- [Вийти](#)

Сезон: 2020/2021

Турнір: Англійська Прем'єр-ліга

Команда: Тоттенгем

Гравець: Гаррі Кейн

Національність: англієць

Ігровий номер: 10

Позиція: нападник

Народження: 28.07.1993

Вартість: 11

Ріст: 188

Вага: 75

Сумарне
Матч

Проведені ігри	11	Відбори	8
Голи	8	Жовті картки	1
Гольові передачі	10	Червоні картки	0
Удари по воротах	42	Порушення	15
Гольові удари	23	Передачі	268

Рисунок 5.3 – Інформація про гравців

Спочатку потрібно обрати сезон з випадаючого переліку, потім турнір в межах обраного сезону. Він також обирається з випадаючого переліку. Далі

з випадваючого переліку «Команда» потрібно обрати одну з команд, що бере участь в турнірі в обраному сезоні. Далі потрібно обрати конкретного гравця цієї команди.

Тоді буде виведено його національність, ігровий номер, позицію, дату народження, вартість (ця вартість є умовною і використовується для виконання оптимального вибору, для того щоб вся команда з 11 гравців коштувала сумарно максимально 100 одиниць), зріст та вага гравця.

Відповідно на вкладках виводиться сумарна та за грою статистика щодо результатів виступу гравця (зокрема кількість ігор, голів, результативних передач тощо).

За допомогою кнопки «Турніри» можна перейти до форми «Таблиця», де за обраним сезоном проведення турніру і назвою турніру буде виведено підсумкову або поточну турнірну таблицю (рис. 5.4).

Моя команда

Гравці

Команди

Найкращі

Профіль

Вийти

Сезон

2020/2021

Турнір

Іспанська Ла Ліга

Місце	Команда	Ігри	В	Н	П	М'ячі	Очки
1	Атлетико	10	8	2	0	21-2	26
2	Реал Сосьєдад	11	7	3	1	22-5	24
3	Вільярреал	12	5	6	1	15-11	21
4	Реал Мадрид	11	6	2	3	17-12	20
5	Кадіс	10	5	3	4	11-13	18
6	Севілья	10	5	1	4	12-9	16
7	Гранада	11	4	3	4	14-20	15
8	Бетис	12	5	0	7	14-23	15
9	Барселона	10	4	2	4	20-11	14
10	Ельче	10	3	5	2	9-10	14
11	Атлетик	11	4	1	6	12-12	13
12	Ейбар	11	3	4	4	8-9	13
13	Алавес	11	3	4	4	11-13	13
14	Гетафе	11	3	4	4	9-13	13
15	Сельта	12	3	4	5	13-20	13
16	Валенсія	11	3	3	5	17-17	12
17	Леванте	11	2	5	4	13-15	11
18	Осасуна	11	3	2	6	8-15	11
19	Вальядолід	12	2	4	6	11-18	10
20	Уеска	12	0	8	4	11-20	8

Рисунок 5.4 – Турнірна таблиця

На відміну від форм адміністратора, форми користувача мають меню, розташоване в лівій частині, що складається з кнопок для відкриття відповідних форм. Склад кнопок змінюється, адже з них зникає та, що відповідає поточній відкритій формі.

Кнопка «Команди» дозволяє переглядати статистику за командами, тобто сумарну статистику за виступом в сезоні команди (на основі статистики всіх гравців) та за виступом в одному матчі (подібно до рис. 5.3).

Кнопка «Найкращі» дозволяє переглядати звіти щодо найкращих гравців за різними критеріями (рис. 5.5).

Моя команда

Гравці

Турніри

Команди

Найкращі команди

Профіль

Вийти

Найкращі гравці

Атакуючі
Комбінаційні
Захисні
Поведінка
Оцінки

Бомбардири

Гравець	Команда	Результат
Роберт Левандовскі	Баварія	12
Ервен Хааланд	Боруссія Дортмунд	10
Ваут Вегхорст	Вольфсбург	7
Лукас Аларіо	Байер	7
Андре Сілва	Айнтрахт	7
Жан-Філіп Матета	Майнц	7
Анджей Крамаріч	Гофенгайм	7
Томас Мюллер	Баварія	6
Матеус Кунья	Герта	6
Макс Крузе	Юніон Берлін	6

Гольові передачі

Гравець	Команда	Результат
Томас Мюллер	Баварія	6
Кінслі Коман	Баварія	5
Даїчі Камада	Айнтрахт	5
Макс Крузе	Юніон Берлін	5
Дані Ольмо	РБ Лейпциг	4
Крістофер Тріммель	Юніон Берлін	4
Флоріан Віртц	Байер	4
Джованні Рейна	Боруссія Дортмунд	4
Йонас Гофман	Боруссія Менхенг...	4
Джошуа Кімміх	Баварія	4

Рисунок 5.5 – Перегляд рейтингів гравців

Всі звіти розділені за вкладками. При переході на цю форму з'являється кнопка «Найкращі команди», що дозволяє переглянути відповідні рейтинги за командами.

Кнопка «Моя команда» дозволяє керувати складом власної команди (рис. 5.6).

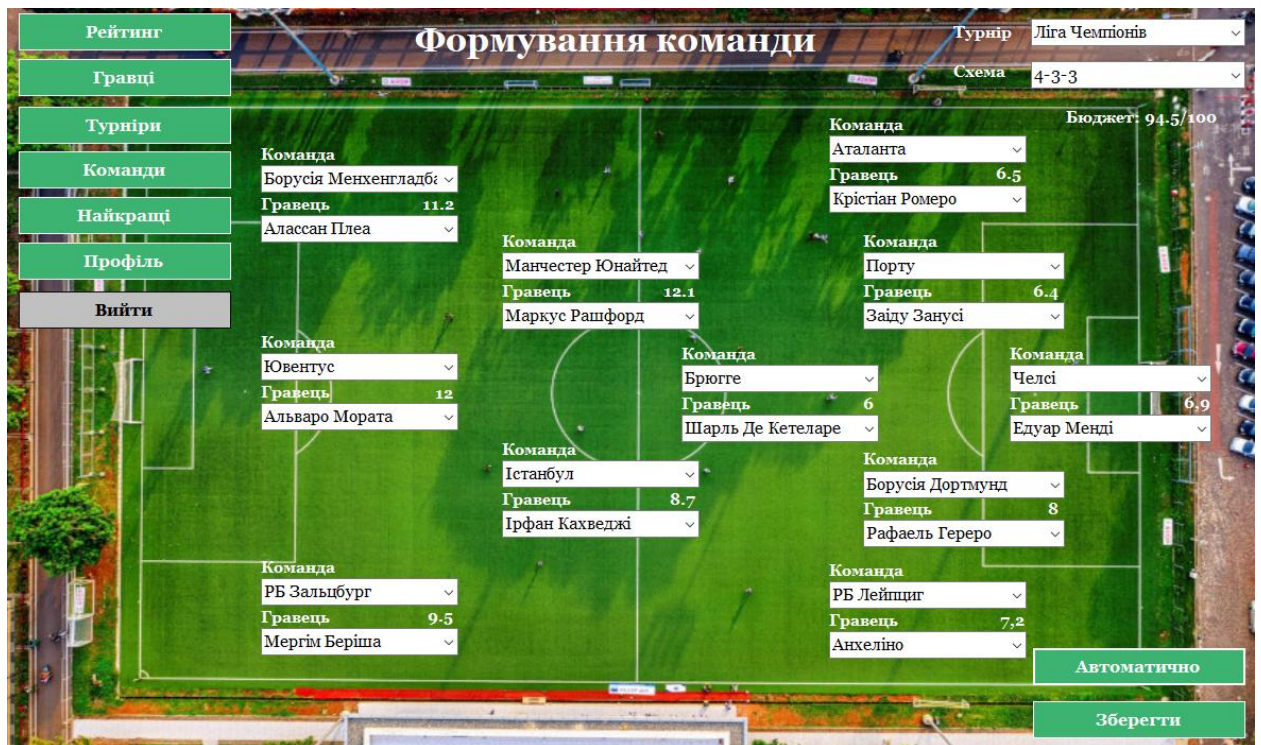


Рисунок 5.6 – Формування складу команди

Перейшовши на цю форму, можна переглянути рейтинг відповідних сформованих іншими користувачами таким чином команд, якщо натиснути на кнопку «Рейтинг».

За даної форми можна обрати турнір (сезон не обирається, оскільки кожен турнір розглядається тільки для поточного сезону) та задати схему, що визначає кількість футболістів у кожній ланці. Для кожної позиції потрібно обрати гравця, обираючи його команду спочатку, а потім ім'я футболіста. Після вибору для кожного гравця відображається його команда, його ім'я та вартість. Окрім того відображається в правому куті загальний бюджет з максимальної суми.

Для автоматичного формування складу команди потрібно натиснути на кнопку «Автоматично».

Щоб зберегти сформовану команду і взяти участь в рейтингу, потрібно натиснути на кнопку «Зберегти».

5.3 Тестування запропонованого методу формування складу футбольної команди

Для тестування запропонованого методу формування складу футбольної команди було використано набір даних, який було сформовано шляхом акумуляції окремих вибірок даних з відкритих джерел [23]-[26]. Зібрані дані було програмно оброблено і внесено в базу даних. Таким чином відбулось початкове заповнення бази даних. У подальшому для роботи програми дані мають вноситися адміністратором.

Тестування відбувалось шляхом розв'язання задачі прогнозування кількості очок, набраних гравцями, в шостому сезоні за результатами виступу в попередніх п'яти.

Для розв'язання задачі було використано три моделі:

- регресійну модель;
- повнозв'язну нейронну мережу;
- просту рекурентну нейронну мережу.

Отримані результати було зведено до табл. 5.1. Оцінювання відбувалось на основі середньоквадратичної помилки.

Таблиця 5.1 – Результати тестування моделей прогнозування

Модель	Середньоквадратична помилка
Регресійна модель	0.1794
Повнозв'язна нейронна мережа	0.0536
Проста рекурентна нейронна мережа	0.0317

Проста рекурентна нейронна мережа продемонструвала найкращий результат. Вона дозволила отримати середньоквадратичну помилку на рівні 0.0317, в той час як повнозв'язна нейронна мережа продемонструвала гірші результати, що вказує на практичну необхідність застосування саме рекурентних нейронних мереж.

В результаті застосування моделі на основі простої рекурентної нейронної мережі для формування складу футбольної команди було отримано помилку в розмірі 6.34 %, тобто сформована на шостий сезон футбольна команда з 11 гравців призвела до отримання кількості очок на 6.34 % меншої за найкращий вибір, який можна було зробити зі всіх наявних за результатами фактичного проведення шостого сезону.

6 ЕКОНОМІКО-ОРГАНІЗАЦІЙНА ЧАСТИНА

6.1 Концепція економічного обґрунтування

Програмне забезпечення призначено для формування складу футбольної команди, що повинно дозволити автоматизувати процес підтримки прийняття рішень під час розв'язання відповідної проблеми. Прийняття рішень полягає у виборі конкретного складу футбольної команди з усіх можливих варіантів. Вибір оптимального складу футбольної команди виконується фахівцем (аналітиком). При цьому введення в експлуатацію програмної системи дозволить фахівцю приймати тільки остаточне рішення, маючи вже обґрунтування можливих варіантів. Відповідно, враховуючи, що фахівець отримує зарплатню за таку роботу, використання програмного забезпечення для автоматизації його роботи дозволить скоротити 0,5 його ставки (з зарплатнею 12 000 грн./міс.).

6.2 Планування розробки програмного продукту

Весь комплекс розробки програмного продукту можна розділити на етапи – для кожного етапу вказуються трудомісткість, кількість виконавців і тривалість робіт [27]. В розробці беруть участь програміст протягом 2,5 місяців і консультант протягом 0,25 місяця.

Тривалість робіт визначають за формулою (6.1):

$$T_{\text{ц}} = \frac{Q}{R}, \quad (6.1)$$

де $T_{\text{ц}}$ – тривалість циклу, днів;

Q – трудомісткість, людино-днів;

R – кількість виконавців, людей [27].

У табл. 6.1 наведена трудомісткість проектування системи, на основі

чого складається зведений стрічковий графік (табл. 6.2) розробки програми, який представляє собою таблицю, в першому стовпці якої розміщені в порядку збільшення строків початку виконання всі види роботи, а в першому рядку – календарний період їх виконання [27]. Зеленим кольором позначена робота програміста, синім – робота консультанта, червоним – святковий день.

Таблиця 6.1 – Характеристика робіт з розробки програми

Найменування роботи	Трудомісткість		Виконавці	Тривалість, дні
	люди-дні	% до підсумку		
Аналіз предметної області	2	3,33	Програміст	2
Узгодження результатів аналізу	3	5	Програміст Консультант	2
Визначення вимог до програмного продукту	4	6,67	Програміст Консультант	3
Узгодження вимог	2	3,33	Програміст	2
Документування вимог	3	5	Програміст	3
Проектування структури програми	15	25	Програміст Консультант	12
Розробка схеми функціонування програми	9	15	Програміст	9
Створення програмного коду	15	25	Програміст	15
Тестування та налагодження програмного продукту	4	6,67	Програміст Консультант	3
Складання програмної документації	3	5	Програміст	3
Підсумок	60	100		54

Таблиця 6.2 – Зведений стрічковий графік планування розробки програмного продукту

Найменування робіт	Календарний період										
	7 – 11.09	14 – 18.09	21 – 25.09	28.09 – 2.10	5 – 9.10	12 – 16.10	19 – 23.10	26 – 30.10	2 – 6.11	9 – 13.11	16 – 20.11
Аналіз предметної області											
Узгодження результатів аналізу	 										
Визначення вимог до програмного продукту		 									
Узгодження вимог											
Документування вимог											
Проектування структури програми			 								
Розробка схеми функціонування програми						 					
Створення програмного коду											
Тестування та налагодження програмного продукту										 	
Складання програмної документації											

6.3 Розрахунок основної заробітної плати

Витрати за цією статтею складаються з планового фонду зарплати всіх категорій працівників, зайнятих у розробці програми (табл. 6.3).

Таблиця 6.3 – Розрахунок основної заробітної плати

Посада виконавця	Чисельність, чол.	Місячний оклад, грн	Кількість місяців роботи	Сума заробітної платні, грн
Програміст	1	10000	2,5	25000
Керівник	1	20000	0,25	5000
Підсумок	2			30000

6.4 Розрахунок додаткової заробітної плати

Додаткову заробітну плату приймають рівною 10% від основної заробітної плати працівників і розраховують за формулою (6.2) [27]:

$$ЗП_{\text{доп}} = ЗП_{\text{осн}} \cdot 0,1. \quad (6.2)$$

У нашому випадку отримаємо:

$$ЗП_{\text{доп}} = 30000 \cdot 0,1 = 3000 \text{ грн.}$$

6.5 Єдиний соціальний внесок

Єдиний соціальний внесок становить 22% і береться від основної та додаткової заробітної плати за формулою (6.3) [27]:

$$ВТ_{\text{соц}} = (ЗП_{\text{осн}} + ЗП_{\text{доп}}) \cdot 0,22; \quad (6.3)$$

$$BT_{\text{сод}} = (30000 + 3000) \cdot 0,22 = 7260 \text{ грн.}$$

6.6 Визначення витрат на матеріали

Використовується 3 найменування матеріалів: диск – 40 грн; тонер для картриджа – 105 грн і папір – 100 грн (1 упаковка).

Витрати на матеріали вираховуються за наступною формулою (6.4):

$$BT_{\text{мат}} = \sum_{i=1}^n (C_i \cdot N_i \cdot (1 + K_{\text{м.з.}}) - C_{\text{іо}} \cdot N_{\text{іо}}), \quad (6.4)$$

де $BT_{\text{мат}}$ – це витрати на матеріали, покупні напівфабрикати та комплектуючі вироби, грн.;

$K_{\text{м.з.}}$ – коефіцієнт, що враховує транспортно–заготівельні витрати;

C_i – ціна i -го найменування матеріалу, напівфабрикату і комплектуючого, грн;

N_i – потреба в i -му матеріалі, напівфабрикати і комплектуючому;

$C_{\text{іо}}$ – ціна зворотних відходів i -го найменування матеріалу, грн;

$N_{\text{іо}}$ – кількість зворотних відходів i -го найменування;

n – кількість найменувань матеріалів, напівфабрикатів і комплектуючих [27].

Розрахуємо:

$$C_{\text{іо}} = 0, N_{\text{іо}} = 0, K_{\text{м.з.}} = 0,05;$$

$$BT_{\text{мат}} = (1 + 0,05) \cdot (40 + 105 + 100) = 257,25 \text{ грн.}$$

Разом, витрати на матеріали становлять 257,25 грн.

6.7 Витрати на спеціальне обладнання

Амортизаційні відрахування визначають за формулою (6.5):

$$A = \Phi_{\phi} \cdot \frac{H_a}{100}, \quad (6.5)$$

де Φ_{ϕ} – балансова вартість обчислювальної техніки, грн;

H_a – норма амортизаційних відрахувань на повне відновлення обчислювальної техніки, % [27].

Норма амортизації для персонального комп'ютера (ПК) становить 25%.

Балансова вартість обчислювальної техніки становить:

$$\Phi_{\phi} = 3000 \text{ грн.}$$

Тоді отримаємо амортизаційні відрахування:

$$A = 3000 \cdot 0,25 = 750 \text{ грн.}$$

Статтю «Експлуатація обладнання» розраховують підсумовуванням витрат на електроенергію та допоміжні матеріали за формулою (6.6):

$$C_e = N_n \cdot \Phi_{ef} \cdot K_{зч} \cdot K_{зн} \cdot \Pi_{еле}, \quad (6.6)$$

де N_n – номінальна потужність електронно-обчислювальної машини (ЕОМ), кВт;

Φ_{ef} – річний ефективний фонд часу роботи ЕОМ, машино-год;

$K_{зч}$ – середній коефіцієнт завантаження по часу;

K_{zn} – коефіцієнт завантаження по потужності;

$C_{ел.е}$ – ціна одного кВт·год електроенергії, грн./(кВт·год) [27].

Номінальна потужність робочої машини – 0,125 кВт/год. Річний ефективний фонд часу роботи ЕОМ становить 1800 годин. Середні коефіцієнти завантаження за часом і по потужності рівні відповідно 0,9 і 0,6. Ціна однієї кіловат–години електроенергії становить 2,11 грн.

Отримуємо ціну експлуатації обладнання:

$$C_e = 0,125 \cdot 1800 \cdot 0,9 \cdot 0,6 \cdot 2,11 = 256,37 \text{ грн.}$$

Зарплата обслуговуючого персоналу розраховується за формулою (6.7):

$$ЗП_{обсл} = \Phi ЗП_p \cdot (1 + K_{відр}) \cdot \frac{t_{обсл}}{\Phi_{эф.обсл}}, \quad (6.7)$$

де $\Phi ЗП_p$ – річний фонд заробітної плати (основної та додаткової) обслуговуючих робітників, грн;

$K_{відр}$ – коефіцієнт, що враховує відрахування на соціальне страхування й в інші фонди;

$t_{обсл}$ – час протягом року, необхідне на технічне обслуговування ЕОМ, год/рік;

$\Phi_{эф.обсл}$ – річний ефективний фонд часу обслуговуючого персоналу, год./рік [27].

Місячна заробітна плата обслуговуючого персоналу становить 3723 грн., а річний фонд заробітної плати відповідно дорівнює 44676 грн. Річний ефективний фонд робочого часу обслуговуючого ПК працівника дорівнює 1750 год/рік. На обслуговування одного ПК витрачається по 1 годині на місяць, що в рік становить 12 годин.

Отримуємо:

$$ЗП_{обсл} = 44676 \cdot (1 + 0,22) \cdot 12 / 1750 = 373,75 \text{ грн.}$$

Стаття «Поточний ремонт обладнання» приймається рівною 3% від балансової вартості обладнання і становить 90 грн.

Стаття «Інші витрати» приймається рівною 5% від суми всіх попередніх статей витрат на утримання і експлуатацію обладнання. Сума всіх попередніх статей дорівнює 1470,12 грн., 5% від суми становлять 73,51 грн.

Розраховані статті витрат на утримання та експлуатацію обладнання внесено до табл. 6.4.

Таблиця 6.4 – Кошторис витрат на утримання та експлуатацію обладнання

Найменування статей витрат	Сума, грн.
Амортизація обладнання	750,00
Експлуатація обладнання (крім витрат на поточний ремонт)	256,37
Заробітна плата основна та додаткова обслуговуючих робітників з відрахуваннями на соціальні заходи	373,75
Поточний ремонт обладнання	90,00
Інші витрати	73,51
Разом	1543,63

Витрати на оплату машинного часу ЕОМ для написання і налагодження даної програми визначаються за формулою (6.8):

$$C_{мо} = B_{екс} \cdot t_{мо}, \quad (6.8)$$

де C_{mo} – витрати на оплату машинного часу, грн;

$B_{екс}$ – експлуатаційні витрати на одну годину машинного часу цієї цифрової ЕОМ, грн/машино–год;

t_{mo} – машинний час ЕОМ для написання і налагодження даного програмного продукту, машино–год. [27].

Експлуатаційні витрати на одну годину машинного часу використовуваної ЕОМ розраховують діленням суми витрат за кошторисом «Витрати на утримання та експлуатацію устаткування (ЕОМ)» (табл. 6.4) на річний ефективний фонд часу роботи ЕОМ. Річний ефективний фонд часу роботи ЕОМ дорівнює 1800 годин. В результаті експлуатаційні витрати на одну годину машинного часу рівні:

$$B_{екс} = 1551,32 / 1800 = 0,86 \text{ грн} / \text{машино} - \text{год}.$$

ЕОМ експлуатується 54 дні в одну зміну у 8 годин, що становить в сумі 432 години. Таким чином, витрати на оплату машинного часу складуть:

$$C_{mo} = 0,86 \cdot 432 = 371,52 \text{ грн}.$$

6.8 Розрахунок накладних витрат

У даному підрозділі проводиться розрахунок накладних витрат.

В даному випадку накладні витрати складають 30% до основної заробітної плати розробників, що становить 9000 грн.

Результати визначення витрат на розробку програми у вигляді калькуляції кошторисної вартості робіт з розробки програми наведені в табл. 6.5.

Таблиця 6.5 – Калькуляція кошторисної вартості робіт з розробки програми

Найменування статей витрат	Сума, грн.	Питома вага до підсумку, %
Основна заробітна плата	30000,00	60,13
Додаткова заробітна плата	3000,00	6,01
Єдиний соціальний внесок	7260,00	14,55
Матеріали та комплектуючі	257,25	0,52
Витрати на спец. обладнання	371,52	0,74
Накладні витрати	9000,00	18,04
Разом	49888,77	100

6.9 Розрахунок економічної ефективності програмного продукту

Програмний продукт дозволяє скоротити час і трудомісткість процесу прийняття рішень. Вибір напрямку з множини альтернатив виконується аналітиком. При цьому введення в експлуатацію програмної системи дозволить фахівцю приймати тільки остаточне рішення, маючи вже обґрунтування можливих варіантів. Відповідно, враховуючи, що аналітик отримує зарплатню за таку роботу, використання програмного забезпечення для автоматизації його роботи дозволить скоротити 0,5 його ставки (з зарплатнею 12 000 грн./міс.).

Річна економія по заробітній платі з урахуванням начислень становить:

$$E_{zn} = 12000 \cdot 6 \cdot (1 + 0,22) = 87840,00 \text{ грн.}$$

Річну економію від впровадження програми, що дозволяє знизити витрати машинного часу ЕОМ для рішення певного завдання в порівнянні із програмою, що застосовувалася раніше, визначають за формулою (6.9):

$$E = (T_{M_1} - T_{M_2}) \cdot B_{\text{екс.КТЗ}} + E_H^a \cdot \frac{\Phi_{\text{б.КТЗ}} \cdot (T_{M_1} - T_{M_2})}{\Phi_{\text{эф.КТЗ}}} - \frac{S_{\text{рп}}}{T_c} + E_{\text{зп}}, \quad (6.9)$$

де T_{M_1}, T_{M_2} – машинний час, необхідний для рішення поставлених завдань, відповідно в старому та у новому варіантах, машино-год./рік;

$B_{\text{екс}}$ – експлуатаційні витрати, що доводяться на 1 год машинного часу ЕОМ, грн/ машино-год.;

E_H^a – нормативний коефіцієнт ефективності капітальних вкладень у засоби автоматизації;

$\Phi_{\text{б.КТЗ}}$ – балансова вартість ЕОМ, грн.;

$\Phi_{\text{эф.КТЗ}}$ – річний ефективний фонд часу роботи ЕОМ, машино-год.;

$S_{\text{рп}}$ – сумарні витрати на розробку програми, грн.;

T_c – термін служби впроваджуваної програми до її морального зношування, років;

$E_{\text{зп}}$ – економія на заробітній платі [27].

Розрахуємо:

$$E = (1800 - 900) \cdot 0,86 + 0,15 \cdot 3000 \cdot (1800 - 900) / 1800 - 49888,77 / 4 + 87840,00 = 76366,81 \text{ грн.}$$

Коефіцієнт ефективності капітальних вкладень E і термін окупності капітальних вкладень $T_{ок}$ розраховуються за формулою (6.10):

$$T_{ок} = S_{pn} / E \text{ [27]}. \quad (6.10)$$

Таким чином:

$$T_{ок} = 49888,77 / 76366,81 = 0,65 \text{ року.}$$

Термін окупності програми становить 0,65 року, що менше нормативного терміну окупності. Річна економія при використанні програми становить 76366,81 грн. Таким чином, впровадження програмного продукту є економічно доцільним.

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

7.1 Аналіз потенційних небезпек

В магістерській роботі проводиться дослідження та програмна реалізація методів формування складу футбольної команди.

При роботі фахівця постійно або періодично діють наступні небезпечні і шкідливі чинники:

- можливість ураження електрострумом внаслідок небезпечного рівня напруги в електричному ланцюзі, тому у приміщені можуть виникнути аварійні ситуації: замикання, загоряння проводки, поразка електрострумом, що може привести до електротравм або летальних наслідків;

- непоінформованість персоналу у відповідній послідовності дій в умовах надзвичайної ситуації, що призводить до травмування та до інших непередбачених негативних наслідків [28].

7.2 Заходи по забезпеченню безпеки

Забезпечення безпеки праці, організації оптимальних умов праці, а також захисту навколишнього середовища досягається шляхом дотримання норм і проведення заходів, передбачених законодавчими актами України, для запобігання підвищеним статичним навантаженням, зокрема, «тунельному синдрому», користувачам ПК рекомендовано:

- для запобігання «тунельного синдрому» робоче місце має бути зручним і ергономічно продуманим;

- використовувати силіконові подушки в якості опори для кисті [29].

Згідно правил улаштування електроустановок (ПУЕ) приміщення виробництва відноситься до категорії «Приміщення без підвищеної небезпеки», при виконанні ПУЕ ураження електричним струмом можливо тільки у випадку несправності апаратури й живильних кабелів: електричний поділ мережі припускає розділ мережі на окремі електрично непов'язані між

собою ділянки за допомогою розділових трансформаторів, підвищення електробезпеки відбувається за рахунок зменшення ємкості мережі й підвищення опору ізоляції фаз мережі щодо землі [28]-[30].

У розділі 4 Державних санітарних правил і норм 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» передбачені гігієнічні вимоги до організації устаткування робочих місць з візуальними дисплейними терміналами ЕОМ і персональних ЕОМ, конструкція робочого стола повинна відповідати сучасним вимогам ергономіки й забезпечувати оптимальне розміщення на робочій поверхні використовуваного устаткування (дисплея, клавіатури, принтера і інших допоміжних пристроїв) і документів [31]-[33].

Велике значення надається ергономічним та естетичним вимогам – цього домоглися за допомогою колірної рішення приміщення й устаткування з урахуванням вимог ГОСТ 17.2.032-78 – у приміщеннях стіни пофарбовані фарбами пастельних тонів: для підвищення продуктивності праці, а також зниження рівня стомлюваності при роботі передбачені десятихвилинні перерви щогодини [34].

7.3 Заходи по забезпеченню виробничої санітарії та гігієни праці

Оптимальні норми температури повітря в робочій зоні приміщення у холодний і перехідний періоди року при II категорії робіт згідно Державних санітарних норм 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» – 20...23 °С, відносна вологість – 60...40%, швидкість руху повітря – 0.2 м/с, у теплий період року температура – 22...25 °С, відносна вологість – 60...40 %, швидкість руху повітря – 0.2 м/с:

– у приміщенні забезпечений приплив свіжого повітря, кількість якого визначається техніко-економічним розрахунком і вибором схеми системи

вентиляції, мінімальна витрата повітря визначається з розрахунку 50...60 м³/год на одну людину, але не менш дворазового повітрообміну в годину;

– при невеликому забрудненні зовнішнього повітря кондиціювання приміщень здійснюється із перемінними витратами зовнішнього повітря й рециркуляційного [35].

Всі норми по освітленості беруться згідно Державних будівельних норм В.2.5-28-2018 «Природне та штучне освітлення»: загальне освітлення варто виконувати у виді суцільних чи переривчастих ліній світильників, розташованих збоку від робочих місць [36].

Розрахуємо систему загального рівномірного штучного освітлення в кабінеті.

Приймаємо розміри приміщення довжина $A = 14$ м, ширина $B = 8$ м, висота $H = 5,6$ м.

Висота робочої поверхні складає $h = 0,7$ м.

Коефіцієнти відбиття поверхонь приміщення: колір стелі, стін – світлий, без пиловиділення, $\rho_c = 70\%$; $\rho_{cm} = 50\%$; $\rho_n = 30\%$.

Розряд зорової роботи – III г.

Згідно з Державними будівельними нормами В.2.5-28-2018 «Природне і штучне освітлення. Норми проектування» рівень нормованого загального освітлення в приміщенні нормуються як $E_n = 200$ Лк.

Обираємо світильник типу ЛПП, призначений для приміщень висотою до 6 м. IP=65 – ступінь захисту світильника, $[L/h] = 1,4$ – числове значення коефіцієнта світильника.

Коефіцієнт запасу k_z враховує зниження рівня освітленості з часом в результаті забруднення та старіння ламп, світильників і поверхонь приміщення, приймається в залежності від виробничих умов. Приймаємо $k_z = 1,6$.

Коефіцієнт нерівномірності (мінімального) освітлення z (відношення середньої освітленості до мінімальної освітленості), для люмінесцентних ламп низького тиску, як правило дорівнює $z = 1,1$.

Розрахуємо кількість рядів світильників у приміщенні за формулою (7.1):

$$N_p = \frac{B}{(H-h_p) \cdot [L/h]}. \quad (7.1)$$

Обчислимо:

$$N_p = \frac{8}{(5,6-0,7) \cdot 1,4} = 1,16 \approx 2 \text{ ряди світильників.}$$

Визначимо максимально припустиму відстань між рядами світильників за формулою (7.2):

$$L_{\max} = \frac{B}{N_p}, \quad (7.2)$$

де B – ширина приміщення.

Обчислимо:

$$L_{\max} = \frac{8}{2} = 4 \text{ м.}$$

Розраховуємо висоту підвісу світильника над робочою поверхнею за формулою (7.3):

$$h = \frac{L_{\max}}{[L/h]}, \quad (7.3)$$

$$h = \frac{4}{1,4} = 2,9 \text{ м.}$$

Знайдемо висоту звисання світильника від стелі за формулою (7.4):

$$h_3 = H - h_p - h, \quad (7.4)$$

де H – висота приміщення;

h_p – висота робочої поверхні;

h – висота підвісу світильника над робочою поверхнею.

Обчислимо:

$$h_3 = 5,6 - 0,7 - 2,9 = 2 \text{ м.}$$

Чисельне значення індексу приміщення визначаємо за рівнянням (7.5):

$$i = \frac{AB}{h(A+B)}, \quad (7.5)$$

де A – довжина приміщення, м;

B – ширина приміщення, м;

h – висота розміщення світильників над робочою поверхнею, м.

Обчислимо:

$$i = \frac{14 \cdot 8}{2,9 \cdot (14 + 8)} = 1,76.$$

Визначаємо сумарний світловий потік освітлювальної установки у даному виробничому приміщенні за формулою (7.6):

$$\Phi_{\Sigma} = \frac{E_n \cdot S \cdot k_3 \cdot z}{\eta}, \quad (7.6)$$

де Φ_{Σ} – розрахункове значення сумарного світлового потоку у приміщенні, лм;

E_n – нормоване значення освітленості, лк;

S – площа освітлюваної поверхні, м²;

k_z – коефіцієнт запасу;

z – коефіцієнт нерівномірності (мінімальної) освітленості;

η – коефіцієнт використання світлового потоку.

Обчислимо:

$$\Phi_{\Sigma} = \frac{200 \cdot 14 \cdot 8 \cdot 1,6 \cdot 1,1}{0,60} = 65706,7 \text{ лм.}$$

Визначаємо умовну загальну кількість світильників у приміщенні за формулою (7.7):

$$N^* = A \cdot \frac{B}{L_{\max}^2}, \quad (7.7)$$

$$N^* = \frac{14 \cdot 8}{4 \cdot 4} = 7 \text{ шт.}$$

Так як $N_p = 2$, то приймаємо $N^* = 8$ шт.

Розрахуємо світловий потік умовного джерела світла за формулами (7.8)-(7.9):

$$\Phi_{\text{л}}^* = \frac{\Phi_{\Sigma}}{N_{\text{л}}}, \quad (7.8)$$

$$N_{\text{л}} = N^* \cdot n, \quad (7.9)$$

де Φ_{Σ} – розрахункове значення сумарного світлового потоку у приміщенні, лм;

$N_{\text{л}}$ – загальна кількість ламп у приміщенні, шт;

n – кількість ламп у світильнику (в ЛПП світильнику знаходиться 2 лампи).

Обчислимо:

$$N_{\text{л}} = 8 \cdot 2 = 16 \text{ шт.}$$

Тоді:

$$\Phi_{\text{л}} = \frac{65706,7}{16} = 4106,7 \text{ лм.}$$

Обираємо лампу ЛД із $P_{\text{л}}=80$ Вт, $\Phi_{\text{л}} = 4300$ лм, за формулою (7.10):

$$m = \frac{\Phi_{\text{л}}^*}{\Phi_{\text{л}}}, \quad (7.10)$$

$$m = \frac{4106,7}{4300} = 0,96.$$

Визначаємо оптимальну (фактичну) кількість світильників у приміщенні за формулою (7.11):

$$N = N^* \cdot m, \quad (7.11)$$

$$N = 8 \cdot 0,96 \approx 8 \text{ шт.}$$

Фактична кількість ламп у приміщенні за формулою (7.12):

$$N_{\text{фл}} = N_{\text{ф}} \cdot n, \quad (7.12)$$

$$N_{\text{фл}} = 8 \cdot 2 = 16 \text{ шт.}$$

Визначаємо загальну розрахункову освітленість E_p у приміщенні, що створюється при застосуванні стандартних ламп, за формулою (7.13):

$$E_p = \frac{\Phi_l N_{\text{фл}} \eta}{Sk_z}, \quad (7.13)$$

$$E_p = \frac{4300 \cdot 16 \cdot 0,6}{14 \cdot 8 \cdot 1,6 \cdot 1,1} = 210 \text{ лк.}$$

Згідно з Державними санітарними нормами 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку» зниження шуму, створюваного на робочих місцях внутрішніми джерелами, а також шуму проникаючого ззовні, є дуже важливим завданням, зниження шуму в джерелі виникнення забезпечується застосуванням пружних прокладок між основою машини, приладу й опорною поверхнею: як прокладки використовуються гума, повсть, корка [37].

Рівні вібрації під час виконання робіт з ЕОМ у виробничих приміщеннях не перевищують допустимих значень, визначених у Державних санітарних нормах і правилах 3.3.2.007-98 «Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» та Державних санітарних нормах 3.3.6-039-99 «Державні санітарні норми виробничої загальної та локальної вібрації» [33], [38].

7.4 Заходи по забезпеченню пожежної безпеки

Заходи щодо протипожежного захисту розділяються на організаційні, експлуатаційні, технічні й режимні (спеціальні), зокрема режимні заходи передбачають заборону або обмеження застосування відкритого вогню в вогнебезпечних місцях, паління в невстановлених місцях, обов'язкове дотримання норм і правил при роботі з вогнебезпечними й вибухонебезпечними речовинами [39].

Клас пожежі для приміщень обчислювальних центрів (ОЦ) встановлюється згідно Державних будівельних нормах В.1.1.7-2002 «Захист від пожежі. Пожежна безпека об'єктів будівництва» – пожежі твердих

речовин, переважно органічного походження, горіння яких супроводжується тлінням (деревина, пластмаси, папір) – і визначається як клас А [40].

Категорія приміщення ОЦ за вибухопожежною безпекою визначається згідно Нормативно-правових актів з пожежної безпеки Б.03.002-2007 «Норми визначення категорії приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною безпекою» – і визначається як пожежонебезпечність адмінприміщень:

- для відводу надлишкового тепла від ЕОМ задіяні кондиціонування повітря і система вентиляції, що обладнана пристроєм, який забезпечує автоматичне відключення вентиляції, в разі виникнення пожежі;

- напруга до електроустановок подається по кабельним мережам, що прокладені під технологічною з'ємною підлогою з рівнем вогнестійкості не менше 0,5 год. [41].

До первинних засобів тушіння пожеж, призначених для ліквідації невеликих загорянь використовуються пожежні стволи, внутрішні пожежні водопроводи, вогнегасники, згідно Нормативно-правових актів з пожежної безпеки Б.03.001-2004 «Типові норми належності вогнегасників» при захисті від пожежі приміщення з ПЕОМ використовуються вуглекислотні вогнегасники ВВК-1,4, ВВК-2 або аерозольні водогінні вогнегасники ВВПА-400 [42].

Коридори та проходи, що призначені для евакуації, мають як можна меншу довжину та мінімальну кількість поворотів, на усій протяжності проходу відсутні пороги або проміжні ступені, відповідно з Санітарними нормами і правилами 2.09.02-85 [43].

7.5 Заходи по забезпеченню безпеки у надзвичайних ситуаціях

Вогнища хімічного ураження (ВХУ) виникають у результаті застосування супротивником отруйних речовин, а також при руйнуванні хімічно небезпечних об'єктів, які виготовляють чи використовують у

технології сильнодіючих отруйних речовин (такі вогнища прийнято називати вторинними): першими у ВХУ для надання допомоги ураженим вводяться медичні підрозділи військових частин і формування медичної служби цивільної оборони [44].

При проведенні рятувальних робіт у вторинному вогнищі ураження основні зусилля спрямовуються на локалізацію джерел отруйної речовини і запобігання її подальшого надходження на місцевість і в повітря: роботи проводяться з максимальною інтенсивністю до повного їх завершення в найкоротший термін, із залученням необхідної кількості сил і засобів [45].

ВИСНОВКИ

За підсумками аналізу предметної області виділено принципи оцінювання виступу футболістів на основі системи очок. Даний показник виділено в якості основи для прогнозування виступу футболістів протягом футбольного турніру.

Наукова новизна роботи полягає в запропонованому методі формування складу футбольної команди на основі оптимізації сумарної оцінки виступу футбольної команди, виконуючи оптимізацію на основі методу динамічного програмування, а прогнозування індивідуальних виступів на основі рекурентних нейронних мереж, що дозволяє в підсумку виконувати автоматизоване прийняття рішень щодо вибору складу футбольної команди з обмеженням щодо вартості.

Для вирішення практичної проблеми було виконано моделювання сценаріїв роботи адміністратора та користувача з програмою, обрано мову C# для реалізації програмного забезпечення, реалізовано визначення структури бази даних за допомогою використання MySQL в якості інструменту роботи з бази даними, визначено структуру програми, описано процес реалізації програми.

Розроблено програмне забезпечення у вигляді настільного застосунку. Програмний застосунок дозволяє накопичувати, переглядати дані футбольної статистики, що включають результати виступу в окремих іграх і накопичені дані за турніром і кар'єрою футболістів, виконувати прогнозування виступу футболістів на турнірі та автоматизовано формувати оптимальний склад команди.

Розроблене програмне забезпечення може використовуватися для керування футбольними командами, отримання доступу до футбольної статистики та організації віртуальних футбольних турнірів.

Усі завдання дипломної кваліфікаційної роботи магістра виконано повністю.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Актуальность футбольной индустрии в мире [Электрон. ресурс]. – Режим доступа : <https://articlekz.com/article/18966>
2. Какие футбольные клубы имеют самые высокие операционные доходы – исследование [Электрон. ресурс]. – Режим доступа : <https://www.epravda.com.ua/rus/news/2020/01/22/656148/>
3. Что такое фэнтези-футбол? [Электрон. ресурс]. – Режим доступа : https://nflrus.ru/content/articles/?e_id=8137&s_id=1
4. Фэнтези-футбол – основные правила [Электрон. ресурс]. – Режим доступа : <https://www.kodpromo.net/fantasy-football>
5. Что такое fantasy football? [Электрон. ресурс]. – Режим доступа : <https://www.sports.ru/tribuna/blogs/ffteam/1365332.html>
6. Fantasy Premier League [Электрон. ресурс] : секреты тех, кто финишировал в 1%. – Режим доступа : <https://baguzin.ru/wp/fantasy-premier-league-sekrety-teh-kto-finishiroval-v-1/>
7. UEFA Champions League – Fantasy Football [Electronic resource]. – Access mode : <https://gaming.uefa.com/en/uclfantasy>
8. Fantasy Football по Лиге чемпионов УЕФА [Электрон. ресурс]. – Режим доступа : <https://gaming.uefa.com/ru/uclfantasy/info/025f-0fe3444021c9-8c634ca32f0c-1000/0>
9. FanTeam – Daily Fantasy & Betting [Electronic resource]. – Access mode : <https://www.fanteam.com/>
10. Football Statistics | Football Live Scores [Electronic resource]. – Access mode : <https://whoscored.com/>
11. Алгоритмы и структуры данных. Методы и парадигмы теории алгоритмов [Электрон. ресурс]. – Режим доступа : <https://dl.sumdu.edu.ua/textbooks/95352/387307/index.html>
12. Алгоритмы [Текст] : построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн. – 2-е изд. – М. : Вильямс, 2005. – 1296 с.

13. Акулич, И. Л. Математическое программирование в примерах и задачах [Текст] / И.Л. Акулич. – М. : Высшая школа, 1986. – 319 с.
14. Artificial neural networks used in optimization problems [Text] / Gabriel Villarrubia, Juan F. De Paz, Pablo Chamoso, Fernando De la Prieta. – Neurocomputing. – 2018. – 272. – P. 10-16.
15. M. Schweidtmann, A. Global Deterministic Optimization with Artificial Neural Networks Embedded [Electronic resource] / Artur M. Schweidtmann, Alexander Mitsos. – Access mode : <https://arxiv.org/pdf/1801.07114.pdf>
16. Интеллектуальные средства диагностики и прогнозирования надежности авиадвигателей [Текст] : Монография / В.И. Дубровин, С.А. Субботин, А.В. Богуслаев, В.К. Яценко. – Запорожье : ОАО «Мотор-Сич», 2003. – 279 с.
17. Субботін, С. О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень [Текст] : навчальний посібник / С.О. Субботін – Запоріжжя : ЗНТУ, 2008. – 341 с.
18. Park, J. Analysis of Recurrent Neural Network and Predictions [Electronic resource] / Jieun Park, Dokkyun Yi and Sangmin Ji. – Symmetry. – 2020. – 12 (615). – Access mode : https://www.researchgate.net/publication/340622781_Analysis_of_Recurrent_Neural_Network_and_Predictions
19. Rahul, P. Pouring Sequence Prediction using Recurrent Neural Network [Electronic resource] / P. Rahul. – Access mode : <https://arxiv.org/ftp/arxiv/papers/1805/1805.09393.pdf>
20. Giles, C. L. Noisy Time Series Prediction using a Recurrent Neural Network and Grammatical Inference [Electronic resource] / C. Lee Giles, Steve Lawrence, Ah Chung Ts. – Access mode : <https://www.cse.unr.edu/~harryt/CS773C/Project/MLJ-finance.pdf>
21. A Formal Hierarchy of RNN Architectures [Electronic resource] / William Merrill, Gail Weiss, Yoav Goldberg et al. // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. – Seattle,

Washington : Association for Computational Linguistics, 2020. – Pp. 443–459. – Access mode : <https://www.aclweb.org/anthology/2020.acl-main.43.pdf>

22. How to implement a simple RNN [Electronic resource]. – Access mode : <https://peterroelants.github.io/posts/rnn-implementation-part01/>

23. Learning and Predicting Sequential Tasks Using Recurrent Neural Networks and Multiple Model Filtering [Electronic resource] / Harish Chaandar Ravichandar, Avnish Kumar, Ashwin P. Dani, Krishna R. Pattipati // The 2016 AAAI Fall Symposium Series. – Access mode : <https://www.aaai.org/ocs/index.php/FSS/FSS16/paper/viewFile/14105/13710>

24. Fantasy Premier League 2019/20(Player Data) [Electronic resource]. – Access mode : <https://www.kaggle.com/plavak10/fantasy-premier-league-201920player-data>

25. English Premier League Players Dataset, 2017/18 [Electronic resource]. – Access mode : <https://www.kaggle.com/mauryashubham/english-premier-league-players-dataset>

26. Fantasy Premier League – 2016/2017 [Electronic resource]. – Access mode : <https://www.kaggle.com/chaibapat/fantasy-premier-league>

27. Методичні вказівки по економічному обґрунтуванню дипломних проектів для студентів спеціальностей 8.05010301 «Програмне забезпечення автоматизованих систем» та 8.05010201 «Інформаційні технології проектування» для усіх форм навчання [Текст] / В.В. Остапенко, Є.М. Сердюк. – Запоріжжя : ЗНТУ, 2015. – 22 с.

28. Жидецький, В. Ц. Основи охорони праці [Текст] : підручник / В. Ц. Жидецький. – 5-те вид., доп. – К. : Знання, 2014. – 373 с.

29. Правила улаштування електроустановок [Текст] : Нормативне виробничо-практичне видання : вид. 5-те, перероблене й доповнене : затв. М-вом енергетики та вугільної промисловості України 20.06.14 : введення в дію з 20.11.14. – Х. : Міненерговугілля України, 2014. – 793 с.

30. ГОСТ 12.1.030-81 «Электробезопасность. Защитное заземление. Зануление» [Текст]. – М. : ИПК Издательство стандартов, 1981. – 8 с.

31. Основи охорони праці [Текст] : підручник / В. В. Березуцький, Т. С. Бондаренко, Г. Г. Валенко та ін. ; за ред. проф. В. В. Березуцького. – Х. : Факт, 2005. – 327 с.
32. ДНАОП 0.00-1.32-01 Правила устройства электроустановок. электрооборудование специальных установок [Текст]. – К. : Киевпромэлектропроект, 2001. – 80 с.
33. ДСанПіН 3.3.2.007-98 Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин [Текст]. – К. : МОЗ, 1998. – 10 с.
34. ГОСТ 17.2.032-78 Рабочее место при выполнении работ сидя Система стандартов безопасности труда [Текст] : Сб. ГОСТов. – М. : ИПК Издательство стандартов, 2001. – 7 с.
35. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень [Текст]. – Харків : Форт, 2000. – 24 с.
36. ДБН В.2.5-28-2006 Природне та штучне освітлення [Текст]. – К. : Мінбуд України, 2006. – 96 с.
37. ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку [Текст]. – К. : МОЗ України, 1999. – 24 с.
38. ДСН 3.3.6-039-99 Державні санітарні норми виробничої загальної та локальної вібрації [Текст]. – К. : МОЗ України, 1999. – 39 с.
39. Катренко, Л. А. Охорона праці. Курс лекцій. Практикум [Текст] : навчальний посібник / Л. А. Катренко, Ю. В. Кіт, І. П. Пісун. – 2-ге вид., стер. – Суми : ВТД «Університетська книга», 2007. – 103 с.
40. ДБН В.1.1.7-2002 Захист від пожежі. Пожежна безпека об'єктів будівництва [Текст]. – К. : Держбуд України, 2003. – 42 с.
41. НАПБ Б.03.002-2007 Норми визначення категорії приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою [Текст]. – К. : МНС України, 2007. – 25 с.
42. НАПБ Б.03.001-2004 Типові норми належності вогнегасників [Текст]. – К. : МНС України, 2004. – 12 с.

43. СНиП 2.09.02-85 Производственные здания [Текст]. – М. : Госстрой, 1991. – 17 с.
44. Шоботов, В. М. Цивільна оборона [Текст] : навчальний посібник / В. М. Шоботов. – Вид. 2-ге, перероб. – К. : Центр навчальної літератури, 2006. – 438 с.
45. Стеблюк, М. І. Цивільна оборона та цивільний захист [Текст] : навч. посіб. для вузів / М. І. Стеблюк. – К. : Знання, 2013. – 487 с.

ДОДАТОК А
Технічне завдання

A.1 Підстави для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу магістра за темою “Дослідження та програмна реалізація методів формування складу футбольної команди”, затверджене наказом № 301 від 3 листопада 2020 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Розробка має виконувати з одного боку збір футбольної статистики за допомогою накопичення широкої статистики відносно кожного матчу кожного турніру, який відслідковується, а з іншого боку надавати можливість виконувати формування власної команди на основі вибору вручну або автоматизованого прийняття відповідного рішення.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Функціональні характеристики, що повинна забезпечувати розробка:

- внесення даних про футбольні турніри, що включає можливості створення нових турнірів, редагування існуючих та за необхідності видалення існуючих;
- внесення даних про футбольні команди, що включає можливості внесення даних про нові команди, редагування існуючих та за необхідності видалення існуючих;
- внесення даних про гравців футбольних команд, зокрема додавання нових гравців, редагування даних про існуючих та за необхідності видалення існуючих;

- додавання футбольних матчів, що проводяться у межах відповідного турніру;
- внесення результатів футбольних матчів, включаючи редагування даних на різних етапах;
- внесення статистичних показників футбольних матчів, зокрема відсотку володіння м'ячем, точності передач команди, кількості нанесених ударів по воротах;
- внесення результатів виступу гравців у футбольних матчах;
- керування набором котирувань;
- внесення даних про фактичні котирування відповідних подій;
- обчислення статистики гравців і результатів команд на основі результатів окремих матчів або турнірів загалом;
- вибір складу власної футбольної команди з можливістю автоматичного формування складу команди;
- прогнозування результатів виступу гравців;
- перегляд найкращих футбольних гравців за обраними користувачем критеріями;
- перегляд найкращих футбольних команд за обраними користувачем критеріями;
- обчислення статистики виступу обраної користувачем футбольної команди;
- відображення результатів проведення футбольного турніру, в тому числі поточних результатів;
- відображення даних гравця загалом та в межах обраного футбольного матчу;
- відображення даних команди загалом та в межах обраного футбольного матчу;
- внесення ставок на події;
- відображення виконаних ставок;

- автоматизоване підведення підсумків за виконаними користувачами ставками;
- авторизація і реєстрація облікових записів.

A.3.2 Умови експлуатації

Програмний продукт повинен використовуватися на комп'ютерах, що працюють під управлінням операційної системи Windows.

A.3.3 Вимоги до складу та параметрів технічних засобів

Мінімальні вимоги до складу та параметрів технічних засобів:

- процесор: 4 ядра, частота – 2 ГГц;
- оперативний запам'ятовуючий пристрій: обсяг – 8 Гб;
- жорсткий диск: обсяг – 100 Гб.

A.3.4 Вимоги до програмної документації

Програмні документи мають охоплювати:

- технічне завдання;
- текст програми.

A.3.5 Вимоги до маркування та упакування

Промарковано фізичний носій має бути назвою «FootballManager».

ДОДАТОК Б
Текст програми

Б.1 Текст файла FootballerStatisticsFrm.cs

```

using System;
using MySql.Data.MySqlClient;
using System.Text;
using System.Collections.Generic;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.ComponentModel;
using System.Windows.Forms.VisualStyles;
using System.Data;
using System.Linq;
using System.Drawing;

namespace FootballManager
{
    public struct ForListElem
    {
        public ForListElem (int v1, string v2)
        {
            id = v1;
            sd = v2;
        }

        public ForListElem ()
        {
            id = 0;
            sd = "";
        }

        public int id { get; private set; }
        public string sd { get; private set; }
    }

    public partial class FootballerStatisticsFrm : Form
    {
        private ConClass conn;
        private int selseasonnum = -1;
        private List <ForListElem> seasonlist = new List
<ForListElem>();
        private int selteamnum = -1;
        private List <ForListElem> teamlist = new List
<ForListElem>();
    }
}

```

```

        private int seltournum = -1;
        private List <ForListElem> tournlist = new List
<ForListElem>();
        private int selffootbnum = -1;
        private List <ForListElem> footnlist = new List
<ForListElem>();
        private int selgamenum = -1;
        private List <ForListElem> gamelist = new List
<ForListElem>();

        public FootballerStatisticsFrm()
        {
            InitializeComponent();
        }

        private void TournamentCB ()
        {
            try
            {
                conn.Create();
                MySqlCommand msc = new MySqlCommand("SELECT
id, name FROM Tournament" +
                "WHERE YEAR(Tournament.tdate) = @ty " +
                "ORDER BY name ", conn.getConn());
                int si = this.selseasonnum;
                ForListElem elem = seasonlist.get(si);
                si = elem.id;
                msc.Parameters.Add("@ty",
MySqlDbType.Int).Value = si;
                MySqlDataReader MSDR = MSC.ExecuteReader();
                if (MSDR.HasRows)
                {
                    while (MSDR.Read())
                    {
                        this.TournamentCB.Items.Add(MSDR["name"]);
                        ForListElem fl;
                        fl.id = MSDR["id"];
                        fl.sd = MSDR["name"];
                        tournlist.add(fl);
                    }
                }
                MSDR.Close();
                this.TournamentCB.SelectedIndex = 0;
                seltournum = -1;
                TeamCB();
            }
            catch { }
        }
    }
}

```

```

        FootballerCB();
    }
    else { conn.Exit(); msdr.Close(); }
    }
    catch { return; }

}

private void TeamCB()
{
    try
    {
        this.AllFTeamsCB.Items.Clear();
        conn.Create();
        MySqlCommand msc = new MySqlCommand("SELECT
Team.name " +
        "FROM Team INNER JOIN TournamentTeam ON
TournamentTeam.team_id = Team.id " +
        "WHERE TournamentTeam.tournament_id = @tid "
+
        "ORDER BY Team.name", conn.getConn());
        msc.Parameters.Add("@tid",
MySQLDbType.Int).Value = this.seltournum;
        MySqlDataReader msdr = msc.ExecuteReader();
        if (msdr.HasRows)
        {
            while (msdr.Read())
            {

this.AllFTeamsCB.Items.Add(msdr["name"]);
                ForListElem fl;
                fl.id = MSDR["id"];
                fl.sd = MSDR["name"];
                teamlist.add(fl);

            }
            conn.Exit();
            msdr.Close();
            this.AllFTeamsCB.SelectedIndex = 0;
            this.selteamnum = -1;
        }
        else { conn.Exit(); msdr.Close(); }
    }
    catch { return; }
}

```

```

private void FootballerCB()
{
    try
    {
        this.AllFootballersCB.Items.Clear();
        if (this.AllFTeamsCB.Items.Count == 0)
        {
            this.AllFootballersCB.Text = "";
            InitFrmParameters();
            return;
        }
        conn.Create();
        MySqlCommand msc = new MySqlCommand("SELECT
Player.name " +
        "FROM Team INNER JOIN Player ON Team.id =
Player.team_id " +
        "WHERE Team.id = @tnum " +
        "ORDER BY Player.name", conn.getConn());
        msc.Parameters.Add("@tnum",
        MySqlDbType.Int).Value = this.selteamnum;
        MySqlDataReader msdr = msc.ExecuteReader();
        if (msdr.HasRows)
        {
            while (msdr.Read())
            {
                this.AllFootballersCB.Items.Add(msdr
["name"]);

                ForListElem fl;
                fl.id = MSDR["id"];
                fl.sd = MSDR["name"];
                footnlist.add(fl);

            }
            conn.Exit();
            msdr.Close();
            this.AllFootballersCB.SelectedIndex = 0;
            this.selffootbnum = -1;
        }
        else { conn.Exit();msdr.Close(); }
    }
    catch { return; }
}

private void GameCB()
{
    try

```

```

    {
        this.AllGamesCB.Items.Clear();
        if (this.AllGamesCB.Items.Count == 0)
        {
            this.AllGamesCB.Text = "";
            InitFrmParameters();
            return;
        }
        conn.Create();
        MySqlCommand msc = new MySqlCommand("SELECT
Game.num AS num, Team.name AS name " +
            "FROM Game INNER JOIN Team ON Game.team1_id =
Team.id AND Game.team2_id = Team.id " +
            "WHERE Game.tournament_id = @tnum AND
(Game.team1_id = @tid OR Game.team2_id = @tid)" +
            "ORDER BY Game.num", conn.getConn());
        msc.Parameters.Add("@tnum",
MySqlDbType.Int).Value = this.seltournum;
        msc.Parameters.Add("@tid",
MySqlDbType.Int).Value = this.selteamnum;
        MySqlDataReader msdr = msc.ExecuteReader();
        if (msdr.HasRows)
        {
            while (msdr.Read())
            {
                String n1 = msdr ["num"].ToString();
                String n2 = msdr ["name"].ToString();
                n2 = n1 + n2;
                this.AllGamesCB.Items.Add(n2);
                ForListElem fl;
                fl.id = MSDR["num"];
                fl.sd = MSDR["name"];
                gamelist.add(fl);
            }
            conn.Exit();
            msdr.Close();
            this.AllGamesCB.SelectedIndex = 0;
            this.selgamenum = -1;
        }
        else { conn.Exit();msdr.Close(); }
    }
    catch { return; }
}

private void InitFrmParameters()

```

```

{
    selteamnum = -1;
    seltournum = -1;
    selfootbnum = -1;
    this.TBFootbNat.Text = "";
    this.TBFootbNum.Text = "";
    this.TBFootbPos.Text = "";
    this.TBFootbBirth.Text = "";
    this.TBFootbHei.Text = "";
    this.TBFootbWei.Text = "";
    this.TBFootbPrice.Text = "";
    this.TBFootbGames.Text = "";
    this.TBFootbGoals.Text = "";
    this.TBFootbGPass.Text = "";
    this.TBFootbAtt.Text = "";
    this.TBFootbGAtt.Text = "";
    this.TBFootbYCard.Text = "";
    this.TBFootbRCard.Text = "";
    this.TBFootbFoals.Text = "";
    this.TBFootbPick.Text = "";
    this.TBFootbPass.Text = "";
}

private void InitFrmParameters2()
{
    selgamenum = -1;
    this.TBFootbGameGoals.Text = "";
    this.TBFootbGameLGoals.Text = "";
    this.TBFootbGameGPasses.Text = "";
    this.TBFootbGameYCards.Text = "";
    this.TBFootbGameRCards.Text = "";
    this.TBFootbGameAtt.Text = "";
    this.TBFootbGameGAtt.Text = "";
    this.TBFootbGameFoals.Text = "";
    this.TBFootbGamePasses.Text = "";
    this.TBFootbGamePicking.Text = "";
    this.TBFootbGamePosition.Text = "";
    this.TBFootbGameMin.Text = "";
    this.TBFootbGamePen.Text = "";
    this.TBFootbGamePenS.Text = "";
    this.TBFootbGamePenF.Text = "";
    this.TBFootbGamePenA.Text = "";
    this.TBFootbGamePenL.Text = "";
    this.TBFootbGameOGoals.Text = "";
    this.TBFootbGameGLost.Text = "";
    this.TBFootbGameSaves.Text = "";
}

```

```

        this.TBFootbGameScores.Text = "";
    }

    private void FootballerStatistics()
    {
        if (AllFootballersCB.Items.Count == 0)
        {
            this.AllFootballersCB.Text = "";
            InitFrmParameters();
            return;
        }
        InitFrmParameters();
        conn.Create();
        MySqlCommand msc = new MySqlCommand("SELECT *
FROM Player " +
        "WHERE Player.id=@pid AND PlayerGame. ",
conn.getConn());

        int si = this.AllFootballersCB.SelectedIndex;
        ForListElem elem = footnlist.get(si);
        si = elem.id;
        msc.Parameters.Add("@pid", MySqlDbType.Int).Value
= si;

        MySqlDataReader msdr = msc.ExecuteReader();
        if (msdr.HasRows)
        {
            while (msdr.Read())
            {
                this.TBFootbNat.Text += msdr ["nation"];
                this.TBFootbNum.Text += msdr ["num"];
                this.TBFootbPos.Text += msdr
["position"];

                this.TBFootbBirth.Text += msdr
["birthday"];

                this.TBFootbHei.Text += msdr ["height"];
                this.TBFootbWei.Text += msdr ["weight"];
                this.TBFootbPrice.Text += msdr ["price"];
                this.TBFootbGames.Text += msdr ["games"];
                this.TBFootbGoals.Text += msdr ["goals"];
                this.TBFootbGPass.Text += msdr
["goalpasses"];

                this.TBFootbAtt.Text += msdr
["attempts"];

                this.TBFootbYCard.Text += msdr
["ycards"];

                this.TBFootbRCard.Text += msdr
["rcards"];
            }
        }
    }

```

```

        this.TBFootbFoals.Text += msdr ["foals"];
        this.TBFootbPick.Text += msdr
["picking"];

        this.TBFootbPass.Text += msdr ["passes"];
        this.TBFootbGAtt.Text += msdr
["goalattempts"];

    }
}
msdr.Close();
conn.closeConnection();
}

private void FootballerGameStatistics
()
{
    if (AllGamesCB.Items.Count == 0)
    {
        this.AllGamesCB.Text = "";
        InitFrmParameters2();
        return;
    }
    InitFrmParameters2();
    conn.Create();
    MySqlCommand msc = new
MySqlCommand("SELECT * FROM PlayerGame " +
        "INNER JOIN Game ON PlayerGame.game_id = Game.id
" +
        "WHERE PlayerGame.player_id=@pid AND
(PlayerGame.team1_id=@tid OR PlayerGame.team2_id=@tid) AND
PlayerGame.num = @nid ", conn.getConn());
    int si = this.AllFootballersCB.SelectedIndex;
    ForListElem elem = footnlist.get(si);
    si = elem.id;
    msc.Parameters.Add("@pid", MySqlDbType.Int).Value
= si;

    si = this.AllTeamsCB.SelectedIndex;
    ForListElem elem = teamlist.get(si);
    si = elem.id;
    msc.Parameters.Add("@tid", MySqlDbType.Int).Value
= si;

    si = this.AllGames.SelectedIndex;
    ForListElem elem = gamelist.get(si);
    si = elem.id;
    msc.Parameters.Add("@nid", MySqlDbType.Int).Value
= si;

    MySqlDataReader msdr = msc.ExecuteReader();
    if (msdr.HasRows)

```

```

{
    while (msdr.Read())
    {
        this.TBFootbGameGoals.Text += msdr
["goals"];
        this.TBFootbGameLGoals.Text += msdr
["longgoals"];
        this.TBFootbGameGPasses.Text += msdr
["goalpasses"];
        this.TBFootbGameYCards.Text += msdr
["ycards"];
        this.TBFootbGameRCards.Text += msdr
["rcards"];
        this.TBFootbGameAtt.Text += msdr
["attempts"];
        this.TBFootbGameGAtt.Text += msdr
["goalattempts"];
        this.TBFootbGameFoals.Text += msdr
["foals"];
        this.TBFootbGamePasses.Text += msdr
["passes"];
        this.TBFootbGamePicking.Text += msdr
["picking"];
        this.TBFootbGamePosition.Text += msdr
["position"];
        this.TBFootbGameMin.Text += msdr
["minutes"];
        this.TBFootbGamePen.Text += msdr
["penalties"];
        this.TBFootbGamePenS.Text += msdr
["pensaved"];
        this.TBFootbGamePenF.Text += msdr
["penfor"];
        this.TBFootbGamePenA.Text += msdr
["penagainst"];
        this.TBFootbGamePenL.Text += msdr
["penlost"];
        this.TBFootbGameOGoals.Text += msdr
["owngoals"];
        this.TBFootbGameGLost.Text += msdr
["goalslost"];
        this.TBFootbGameSaves.Text += msdr
["saves"];
        this.TBFootbGameScores.Text += msdr
["scores"];
    }
}

```

```

    }
    msdr.Close();
    conn.Exit();
}

private void FootballerStatisticsInitFrm(object
sender, EventArgs e)
{
    conn = new ConClass();
    try
    {
        conn.Create();
        MySqlCommand msc = new MySqlCommand("SELECT
DISTINCT YEAR(tdate) AS Year FROM Tournament" +
"ORDER BY Year ", conn.getConnection());
        MySqlDataReader MSDR = MSC.ExecuteReader();
        if (MSDR.HasRows)
        {
            while (MSDR.Read())
            {
                int y = MSDR["Year"].ToInt32();
                int y2 = y-1;

                this.SeasonCB.Items.Add(y2.ToString()+"/"+y.ToString());
                ForListElem fl;
                fl.id = MSDR["Year"];
                fl.sd = MSDR["Year"];
                seasonlist.add(fl)
            }
        }
        MSDR.Close();
        this.SeasonCB.SelectedIndex = 0;
        seltournum = -1;
        TournamentCB();
        TeamCB();
        FootballerCB();
        FootballerGameStatistics();
    }
    catch
    {
        MessageBox.Show("Не вдалося задати турнір",
"Проблема", MessageBoxButtons.OK, MessageBoxIcon.Error);
        conn.Exit();
        this.TournamentCB.Text = "";
        this.AllFTeamsCB.Text = "";
        this.AllFootballersCB.Text = "";
    }
}

```

```

        InitFrmParameters();
        return;
    }
}

private void AllFootballersCBSI(object sender,
EventArgs e)
{
    int si = this.AllFTeamsCB.SelectedIndex;
    ForListElem elem = this.footnlist.get(si);
    this.selfootbnum = elem.id;
    GameCB();
    FootballerStatistics();
    FootballerGameStatistics();
}

private void AllGamesCBSI(object sender, EventArgs e)
{
    int si = this.AllGamesCB.SelectedIndex;
    ForListElem elem = this.gamelist.get(si);
    this.selgamenum = elem.id;
    FootballerGameStatistics();
}

private void AllFTeamsCBSI(object sender, EventArgs
e)
{
    int si = this.AllFTeamsCB.SelectedIndex;
    ForListElem elem = this.teamlist.get(si);
    this.selteamnum = elem.id;
    FootballerCB();
    FootballerStatistics();
    FootballerGameStatistics();
}

private void SeasonCBSI(object sender, EventArgs e)
{
    this.AllSeasonsCB.Text = "";
    this.selseasonnum = this.SeasonCB.Text.ToInt32();
    TournamentCB();
    TeamCB();
    FootballerCB();
    FootballerStatistics();
    FootballerGameStatistics();
}

```

```

private void TournamentCBSI(object sender, EventArgs
e)
{
    this.AllTournamentsCB.Text = "";
    int si = this.TournamentCB.SelectedIndex;
    ForListElem elem = this.tournlist.get(si);
    this.seltournum = elem.id;
    TeamCB();
    FootballerCB();
    FootballerStatistics();
    FootballerGameStatistics();
}

private void OutBtnClick(object sender, EventArgs e)
{
    Authorize.Leave();
    this.Close();
    Application.Exit();
}

private void MyTeamBtnClick(object sender, EventArgs
e)
{
    MyTeamFrm mt = new MyTeamFrm ();
    mt.Show();
    this.Close();
}

private void TournamentsBtnClick(object sender,
EventArgs e)
{
    TournamentsFrm tf = new TournamentsFrm ();
    tf.Show();
    this.Close();
}

private void TeamsBtnClick(object sender, EventArgs
e)
{
    TeamsFrm tf = new TeamsFrm ();
    tf.Show();
    this.Close();
}

private void ProfileBtnClick(object sender, EventArgs
e)

```

```

        {
            ProfileFrm pf = new ProfileFrm ();
            pf.Show();
            this.Close();
        }

private void BestPlBtnClick(object sender, EventArgs
e)
    {
        BestPlFrm pf = new BestPlFrm ();
        pf.Show();
        this.Close();
    }

private void OutBtnClick(object sender, EventArgs e)
    {
        DialogResult dr = MessageBox.Show("Точно вийти з
облікового запису?", "Розавторизація", MessageBoxButtons.YesNo,
MessageBoxIcon.Question);
        if (dr == DialogResult.No) return;
        else
        {
            Authorize.Leave();
            AuthorizeFrm af = new AuthorizeFrm();
            af.Show();
            this.Close();
        }
    }
}

```

Б.2 Текст файлу BestPlFrm.cs

```

using System;
using System.Linq;
using System.Collections.Generic;
using System.Data;
using System.Text;
using System.ComponentModel;
using MySql.Data.MySqlClient;
using System.Windows.Forms;
using System.Drawing;
using System.Threading.Tasks;

```

```

namespace FootballManager
{
    public struct ResultListElem
    {
        public ResultListElem (int v1, float v2, string v3,
string v4)
        {
            id = v1;
            fd = v2;
            name = v3;
            team = v4;
        }

        public ResultListElem ()
        {
            id = 0;
            fd = 0;
            name = "";
            team = "";
        }
    }

    public int id { get; private set; }
    public string name { get; private set; }
    public string team { get; private set; }
    public float fd { get; private set; }
}

public partial class BestPlFrm : Form
{
    private ConClass conn;
    private int selseasonnum = -1;
    private List <ResultListElem> seasonlist = new List
<ResultListElem>();
    private int seltournum = -1;
    private List <ResultListElem> tournlist = new List
<ResultListElem>();

    public BestPlFrm ()
    {
        InitializeComponent();
    }

    private void OutBtnClick (object sender, EventArgs e)
    {
        Authorize.ExitUser();
        this.Close();
        Application.Exit();
    }
}

```

```

    }

    private void TournamentCB ()
    {
        try
        {
            conn.Create();
            MySqlCommand msc = new MySqlCommand("SELECT
id, name FROM Tournament" +
            "WHERE YEAR(Tournament.tdate) = @ty " +
            "ORDER BY name ", conn.getConn());
            int si = this.selseasonnum;
            ResultListElem elem = seasonlist.get(si);
            si = elem.id;
            msc.Parameters.Add("@ty",
MySqlDbType.Int).Value = si;
            MySqlDataReader MSDR = MSC.ExecuteReader();
            if (MSDR.HasRows)
            {
                while (MSDR.Read())
                {

                    this.TournamentCB.Items.Add(MSDR["name"]);
                    ResultListElem fl;
                    fl.id = MSDR["id"];
                    fl.sd = MSDR["name"];
                    tournlist.add(fl);

                }
            }
            MSDR.Close();
            this.TournamentCB.SelectedIndex = 0;
            seltournum = -1;
            TeamCB();
            FootballerCB();
        }
        else { conn.Exit(); msdr.Close(); }
        catch { return; }
    }

    private void BestPlFrmInit (object sender, EventArgs
e)
    {
        conn = new ConClass ();
        try

```

```

        {
            conn.Create();
            MySqlCommand msc = new MySqlCommand("SELECT
DISTINCT YEAR(tdate) AS Year FROM Tournament" +
            "ORDER BY Year ", conn.getConn());
            MySqlDataReader MSDR = MSC.ExecuteReader();
            if (MSDR.HasRows)
            {
                while (MSDR.Read())
                {
                    int y = MSDR["Year"].ToInt32();
                    int y2 = y-1;

                    this.SeasonCB.Items.Add(y2.ToString()+"/"+y.ToString());
                    ResultListElem fl;
                    fl.id = MSDR["Year"];
                    fl.sd = MSDR["Year"];
                    seasonlist.add(fl)
                }
            }
            MSDR.Close();
            this.SeasonCB.SelectedIndex = 0;
            seltournum = -1;
            TournamentCB();
            BestInfo();
        }
        catch
        {
            MessageBox.Show("Не вдалося задати турнір",
"Проблема", MessageBoxButtons.OK, MessageBoxIcon.Error);
            conn.Exit();
            this.TournamentCB.Text = "";
            return;
        }
    }

private void SeasonCBSI(object sender, EventArgs e)
{
    this.AllSeasonsCB.Text = "";
    this.selseasonnum = this.SeasonCB.Text.ToInt32();
    TournamentCB();
    BestInfo();
}

```

```

private void TournamentCBSI(object sender, EventArgs
e)
{
    this.AllTournamentsCB.Text = "";
    int si = this.TournamentCB.SelectedIndex;
    ResultListElem elem = this.tournlist.get(si);
    this.seltournum = elem.id;
    BestInfo ();
}

private void BestInfo()
{
    int curr;
    MySqlCommand msc, msc2, msc3;
    MySqlDataReader msdr, msdr2, msdr3;
    List<ResultListElem> results = new
List<ResultListElem>(), resGPASSES = new List<ResultListElem>(),
resYCards= new List<ResultListElem>(), resRCards = new
List<ResultListElem>(), resAtt = new List<ResultListElem>(),
resGAtt = new List<ResultListElem>(), resFoals = new
List<ResultListElem>(), resPasses = new List<ResultListElem>(),
resPick = new List<ResultListElem>(), resMins = new
List<ResultListElem>(), resPens = new List<ResultListElem>(),
resPenSav = new List<ResultListElem>(), resSaves = new
List<ResultListElem>(), resScores = new List<ResultListElem>();
    this.dgGoals.Rows.Clear();
    this.dgGPASSES.Rows.Clear();
    this.dgYCards.Rows.Clear();
    this.dgRCards.Rows.Clear();
    this.dgAtt.Rows.Clear();
    this.dgGAtt.Rows.Clear();
    this.dgFoals.Rows.Clear();
    this.dgPasses.Rows.Clear();
    this.dgPick.Rows.Clear();
    this.dgMins.Rows.Clear();
    this.dgPens.Rows.Clear();
    this.dgPenSav.Rows.Clear();
    this.dgSaves.Rows.Clear();
    this.dgScores.Rows.Clear();
    conn.Create();
    msc = new MySqlCommand("SELECT DISTINCT
PlayerGame.player_id AS pd FROM PlayerGame INNER JOIN Game ON
Game.id = PlayerGame.game_id WHERE Game.tournament_id = @tid",
conn.getConn());
    msc.Parameters.Add("@tid ",
MySqlDbType.Int32).Value = this.seltournum;

```

```

msdr = msc.ExecuteReader();
if (msdr.HasRows)
{
    while (msdr.Read())
    {
        curr = msdr["pd"];
        msc2 = new MySqlCommand("SELECT
SUM(PlayerGame.goals) AS pg FROM PlayerGame INNER JOIN Game ON
Game.id = PlayerGame.game_id WHERE PlayerGame.played_id = @pid AND
Game.tournament_id ", conn.getConnection());
        msc2.Parameters.Add("@tid ",
MySqlDbType.Int32).Value = this.seltournum;
        msc2.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;
        msdr2 = msc2.ExecuteReader();
        if (msdr2.HasRows)
        {
            while (msdr2.Read())
            {
                msc3 = new MySqlCommand("SELECT
Player.name AS pn, Team.name AS tn FROM Player INNER JOIN Team ON
Player.team_id = Team.id WHERE Player.id = @pid", conn.getConnection());
                msc3.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;

                msdr3 = msc3.ExecuteReader();
                if (msdr3.HasRows)
                {
                    while (msdr3.Read())
                    {

                        ResultListElem fl;
                        fl.name =

msdr3["pn"];

                        fl.id = curr;
                        fl.team =

msdr3["tn"];

                        fl.fd = msdr2["pg"];
                        results.add(fl);

                    }
                }
                msdr3.Close();
            }
        }
        msdr2.Close();
        msc2 = new MySqlCommand("SELECT
SUM(PlayerGame.goalpasses) AS pg FROM PlayerGame INNER JOIN Game

```

```

ON Game.id = PlayerGame.game_id WHERE PlayerGame.played_id = @pid
AND Game.tournament_id ", conn.getConn());
        msc2.Parameters.Add("@tid ",
MySqlDbType.Int32).Value = this.seltournum;
        msc2.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;
        msdr2 = msc2.ExecuteReader();
        if (msdr2.HasRows)
        {
            while (msdr2.Read())
            {
                msc3 = new MySqlCommand("SELECT
Player.name AS pn, Team.name AS tn FROM Player INNER JOIN Team ON
Player.team_id = Team.id WHERE Player.id = @pid", conn.getConn());
                msc3.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;

                msdr3 = msc3.ExecuteReader();
                if (msdr3.HasRows)
                {
                    while (msdr3.Read())
                    {

                        ResultListElem fl;
                        fl.name =

msdr3["pn"];

                        fl.id = curr;
                        fl.team =

msdr3["tn"];

                        fl.fd = msdr2["pg"];
                        resGPasses.add(fl);

                    }
                }
                msdr3.Close();
            }
        }
        msdr2.Close();
        msc2 = new MySqlCommand("SELECT
SUM(PlayerGame.ycards) AS pg FROM PlayerGame INNER JOIN Game ON
Game.id = PlayerGame.game_id WHERE PlayerGame.played_id = @pid AND
Game.tournament_id ", conn.getConn());
        msc2.Parameters.Add("@tid ",
MySqlDbType.Int32).Value = this.seltournum;
        msc2.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;
        msdr2 = msc2.ExecuteReader();
        if (msdr2.HasRows)

```

```

        {
            while (msdr2.Read())
            {
                msc3 = new MySqlCommand("SELECT
Player.name AS pn, Team.name AS tn FROM Player INNER JOIN Team ON
Player.team_id = Team.id WHERE Player.id = @pid", conn.getConn());
                msc3.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;
                msdr3 = msc3.ExecuteReader();
                if (msdr3.HasRows)
                {
                    while (msdr3.Read())
                    {
                        ResultListElem fl;
                        fl.name =
msdr3["pn"];
                        fl.id = curr;
                        fl.team =
msdr3["tn"];
                        fl.fd = msdr2["pg"];
                        resYCards.add(fl);
                    }
                }
                msdr3.Close();
            }
            msdr2.Close();
            msc2 = new MySqlCommand("SELECT
SUM(PlayerGame.rcards) AS pg FROM PlayerGame INNER JOIN Game ON
Game.id = PlayerGame.game_id WHERE PlayerGame.played_id = @pid AND
Game.tournament_id ", conn.getConn());
            msc2.Parameters.Add("@tid ",
MySqlDbType.Int32).Value = this.seltournum;
            msc2.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;
            msdr2 = msc2.ExecuteReader();
            if (msdr2.HasRows)
            {
                while (msdr2.Read())
                {
                    msc3 = new MySqlCommand("SELECT
Player.name AS pn, Team.name AS tn FROM Player INNER JOIN Team ON
Player.team_id = Team.id WHERE Player.id = @pid", conn.getConn());
                    msc3.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;

```

```

msdr3 = msc3.ExecuteReader();
if (msdr3.HasRows)
{
    while (msdr3.Read())
    {
        ResultListElem fl;
        fl.name =

msdr3["pn"];

        fl.id = curr;
        fl.team =

msdr3["tn"];

        fl.fd = msdr2["pg"];
        resRCards.add(fl);
    }
}
msdr3.Close();
}
}
msdr2.Close();
msc2 = new MySqlCommand("SELECT
SUM(PlayerGame.attempts) AS pg FROM PlayerGame INNER JOIN Game ON
Game.id = PlayerGame.game_id WHERE PlayerGame.played_id = @pid AND
Game.tournament_id ", conn.getConnection());
msc2.Parameters.Add("@tid ",
MySqlDbType.Int32).Value = this.seltournum;
msc2.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;
msdr2 = msc2.ExecuteReader();
if (msdr2.HasRows)
{
    while (msdr2.Read())
    {
        msc3 = new MySqlCommand("SELECT
Player.name AS pn, Team.name AS tn FROM Player INNER JOIN Team ON
Player.team_id = Team.id WHERE Player.id = @pid", conn.getConnection());
        msc3.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;

        msdr3 = msc3.ExecuteReader();
        if (msdr3.HasRows)
        {
            while (msdr3.Read())
            {

                ResultListElem fl;

```

```

msdr3["pn"];

msdr3["tn"];

fl.name =

fl.id = curr;
fl.team =

fl.fd = msdr2["pg"];
resAtt.add(fl);
    }
    }
    msdr3.Close();
    }
    }
    msdr2.Close();
    msc2 = new MySqlCommand("SELECT
SUM(PlayerGame.goalattempts) AS pg FROM PlayerGame INNER JOIN Game
ON Game.id = PlayerGame.game_id WHERE PlayerGame.played_id = @pid
AND Game.tournament_id ", conn.getConnection());
    msc2.Parameters.Add("@tid ",
MySqlDbType.Int32).Value = this.seltournum;
    msc2.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;
    msdr2 = msc2.ExecuteReader();
    if (msdr2.HasRows)
    {
        while (msdr2.Read())
        {
            msc3 = new MySqlCommand("SELECT
Player.name AS pn, Team.name AS tn FROM Player INNER JOIN Team ON
Player.team_id = Team.id WHERE Player.id = @pid", conn.getConnection());
            msc3.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;

            msdr3 = msc3.ExecuteReader();
            if (msdr3.HasRows)
            {
                while (msdr3.Read())
                {

ResultListElem fl;
fl.name =

fl.id = curr;
fl.team =

fl.fd = msdr2["pg"];
resGAtt.add(fl);
                }
            }
        }
    }

```

```

        }
        msdr3.Close();
    }
}
msdr2.Close();
msc2 = new MySqlCommand("SELECT
SUM(PlayerGame.foals) AS pg FROM PlayerGame INNER JOIN Game ON
Game.id = PlayerGame.game_id WHERE PlayerGame.played_id = @pid AND
Game.tournament_id ", conn.getConn());
msc2.Parameters.Add("@tid ",
MySqlDbType.Int32).Value = this.seltournum;
msc2.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;
msdr2 = msc2.ExecuteReader();
if (msdr2.HasRows)
{
    while (msdr2.Read())
    {
        msc3 = new MySqlCommand("SELECT
Player.name AS pn, Team.name AS tn FROM Player INNER JOIN Team ON
Player.team_id = Team.id WHERE Player.id = @pid", conn.getConn());
        msc3.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;
        msdr3 = msc3.ExecuteReader();
        if (msdr3.HasRows)
        {
            while (msdr3.Read())
            {
                ResultListElem fl;
                fl.name =

msdr3["pn"];

                fl.id = curr;
                fl.team =

msdr3["tn"];

                fl.fd = msdr2["pg"];
                resFoals.add(fl);
            }
        }
        msdr3.Close();
    }
}
msdr2.Close();
msc2 = new MySqlCommand("SELECT
SUM(PlayerGame.passes) AS pg FROM PlayerGame INNER JOIN Game ON

```

```

Game.id = PlayerGame.game_id WHERE PlayerGame.played_id = @pid AND
Game.tournament_id ", conn.getConn());
        msc2.Parameters.Add("@tid ",
MySqlDbType.Int32).Value = this.seltournum;
        msc2.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;
        msdr2 = msc2.ExecuteReader();
        if (msdr2.HasRows)
        {
            while (msdr2.Read())
            {
                msc3 = new MySqlCommand("SELECT
Player.name AS pn, Team.name AS tn FROM Player INNER JOIN Team ON
Player.team_id = Team.id WHERE Player.id = @pid", conn.getConn());
                msc3.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;

                msdr3 = msc3.ExecuteReader();
                if (msdr3.HasRows)
                {
                    while (msdr3.Read())
                    {

                        ResultListElem fl;
                        fl.name =

msdr3["pn"];

                        fl.id = curr;
                        fl.team =

msdr3["tn"];

                        fl.fd = msdr2["pg"];
                        resPasses.add(fl);

                    }
                }
                msdr3.Close();
            }
        }
        msdr2.Close();
        msc2 = new MySqlCommand("SELECT
SUM(PlayerGame.picking) AS pg FROM PlayerGame INNER JOIN Game ON
Game.id = PlayerGame.game_id WHERE PlayerGame.played_id = @pid AND
Game.tournament_id ", conn.getConn());
        msc2.Parameters.Add("@tid ",
MySqlDbType.Int32).Value = this.seltournum;
        msc2.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;
        msdr2 = msc2.ExecuteReader();
        if (msdr2.HasRows)

```

```

        {
            while (msdr2.Read())
            {
                msc3 = new MySqlCommand("SELECT
Player.name AS pn, Team.name AS tn FROM Player INNER JOIN Team ON
Player.team_id = Team.id WHERE Player.id = @pid", conn.getConn());
                msc3.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;

                msdr3 = msc3.ExecuteReader();
                if (msdr3.HasRows)
                {
                    while (msdr3.Read())
                    {

                        ResultListElem fl;
                        fl.name =

msdr3["pn"];

                        fl.id = curr;
                        fl.team =

msdr3["tn"];

                        fl.fd = msdr2["pg"];
                        resPick.add(fl);

                    }
                }
                msdr3.Close();
            }
        }
        msdr2.Close();
        msc2 = new MySqlCommand("SELECT
SUM(PlayerGame.minutes) AS pg FROM PlayerGame INNER JOIN Game ON
Game.id = PlayerGame.game_id WHERE PlayerGame.played_id = @pid AND
Game.tournament_id ", conn.getConn());
        msc2.Parameters.Add("@tid ",
MySqlDbType.Int32).Value = this.seltournum;
        msc2.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;
        msdr2 = msc2.ExecuteReader();
        if (msdr2.HasRows)
        {
            while (msdr2.Read())
            {
                msc3 = new MySqlCommand("SELECT
Player.name AS pn, Team.name AS tn FROM Player INNER JOIN Team ON
Player.team_id = Team.id WHERE Player.id = @pid", conn.getConn());
                msc3.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;

```

```

msdr3 = msc3.ExecuteReader();
if (msdr3.HasRows)
{
    while (msdr3.Read())
    {
        ResultListElem fl;
        fl.name =

msdr3["pn"];

        fl.id = curr;
        fl.team =

msdr3["tn"];

        fl.fd = msdr2["pg"];
        resMins.add(fl);
    }
}
msdr3.Close();
}
}
msdr2.Close();
msc2 = new MySqlCommand("SELECT
SUM(PlayerGame.penalties) AS pg FROM PlayerGame INNER JOIN Game ON
Game.id = PlayerGame.game_id WHERE PlayerGame.played_id = @pid AND
Game.tournament_id ", conn.getConnection());
msc2.Parameters.Add("@tid ",
MySqlDbType.Int32).Value = this.seltournum;
msc2.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;
msdr2 = msc2.ExecuteReader();
if (msdr2.HasRows)
{
    while (msdr2.Read())
    {
        msc3 = new MySqlCommand("SELECT
Player.name AS pn, Team.name AS tn FROM Player INNER JOIN Team ON
Player.team_id = Team.id WHERE Player.id = @pid", conn.getConnection());
        msc3.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;

        msdr3 = msc3.ExecuteReader();
        if (msdr3.HasRows)
        {
            while (msdr3.Read())
            {

                ResultListElem fl;

```

```

msdr3["pn"];

msdr3["tn"];

fl.name =

fl.id = curr;
fl.team =

fl.fd = msdr2["pg"];
resPens.add(fl);
    }
    }
    msdr3.Close();
    }
    }
    msdr2.Close();
    msc2 = new MySqlCommand("SELECT
SUM(PlayerGame.pensaved) AS pg FROM PlayerGame INNER JOIN Game ON
Game.id = PlayerGame.game_id WHERE PlayerGame.played_id = @pid AND
Game.tournament_id ", conn.getConnection());
    msc2.Parameters.Add("@tid ",
MySqlDbType.Int32).Value = this.seltournum;
    msc2.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;
    msdr2 = msc2.ExecuteReader();
    if (msdr2.HasRows)
    {
        while (msdr2.Read())
        {
            msc3 = new MySqlCommand("SELECT
Player.name AS pn, Team.name AS tn FROM Player INNER JOIN Team ON
Player.team_id = Team.id WHERE Player.id = @pid", conn.getConnection());
            msc3.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;

            msdr3 = msc3.ExecuteReader();
            if (msdr3.HasRows)
            {
                while (msdr3.Read())
                {

ResultListElem fl;
fl.name =

fl.id = curr;
fl.team =

fl.fd = msdr2["pg"];
resPenSav.add(fl);
                }
            }
        }
    }

```

```

        }
        msdr3.Close();
    }
}
msdr2.Close();
msc2 = new MySqlCommand("SELECT
SUM(PlayerGame.saves) AS pg FROM PlayerGame INNER JOIN Game ON
Game.id = PlayerGame.game_id WHERE PlayerGame.played_id = @pid AND
Game.tournament_id ", conn.getConn());
    msc2.Parameters.Add("@tid ",
MySqlDbType.Int32).Value = this.seltournum;
    msc2.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;
    msdr2 = msc2.ExecuteReader();
    if (msdr2.HasRows)
    {
        while (msdr2.Read())
        {
            msc3 = new MySqlCommand("SELECT
Player.name AS pn, Team.name AS tn FROM Player INNER JOIN Team ON
Player.team_id = Team.id WHERE Player.id = @pid", conn.getConn());
            msc3.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;

            msdr3 = msc3.ExecuteReader();
            if (msdr3.HasRows)
            {
                while (msdr3.Read())
                {

                    ResultListElem fl;
                    fl.name =

msdr3["pn"];

                    fl.id = curr;
                    fl.team =

msdr3["tn"];

                    fl.fd = msdr2["pg"];
                    resSaves.add(fl);

                }
            }
            msdr3.Close();
        }
    }
    msdr2.Close();
    msc2 = new MySqlCommand("SELECT
SUM(PlayerGame.scores) AS pg FROM PlayerGame INNER JOIN Game ON

```

```

Game.id = PlayerGame.game_id WHERE PlayerGame.played_id = @pid AND
Game.tournament_id ", conn.getConn());
        msc2.Parameters.Add("@tid ",
MySqlDbType.Int32).Value = this.seltournum;
        msc2.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;
        msdr2 = msc2.ExecuteReader();
        if (msdr2.HasRows)
        {
            while (msdr2.Read())
            {
                msc3 = new MySqlCommand("SELECT
Player.name AS pn, Team.name AS tn FROM Player INNER JOIN Team ON
Player.team_id = Team.id WHERE Player.id = @pid", conn.getConn());
                msc3.Parameters.Add("@pid ",
MySqlDbType.Int32).Value = curr;

                msdr3 = msc3.ExecuteReader();
                if (msdr3.HasRows)
                {
                    while (msdr3.Read())
                    {

                        ResultListElem fl;
                        fl.name =

msdr3["pn"];

                        fl.id = curr;
                        fl.team =

msdr3["tn"];

                        fl.fd = msdr2["pg"];
                        resScores.add(fl);

                    }
                }
                msdr3.Close();
            }
        }
        msdr2.Close();
    }
}

List<ResultListElem> gslist =
results.OrderByDescending(o=>o.fd).ToList();
for (int i = 0; i < 10; i++)
{
    this.dgGoals.Rows.Add(gslist [0].name, gslist
[0].team, gslist [0].fd;
}

```

```

        gslist =
resGPasses.OrderByDescending(o=>o.fd).ToList();
        for (int i = 0; i < 10; i++)
        {
            this.dgGPasses.Rows.Add(gslist [0].name, gslist
[0].team, gslist [0].fd;
        }
        gslist =
resYCards=.OrderByDescending(o=>o.fd).ToList();
        for (int i = 0; i < 10; i++)
        {
            this.dgYCards.Rows.Add(gslist [0].name, gslist
[0].team, gslist [0].fd;
        }
        gslist =
resRCards.OrderByDescending(o=>o.fd).ToList();
        for (int i = 0; i < 10; i++)
        {
            this.dgRCards.Rows.Add(gslist [0].name, gslist
[0].team, gslist [0].fd;
        }
        gslist =
resAtt.OrderByDescending(o=>o.fd).ToList();
        for (int i = 0; i < 10; i++)
        {
            this.dgAtt.Rows.Add(gslist [0].name, gslist
[0].team, gslist [0].fd;
        }
        gslist =
resGAtt.OrderByDescending(o=>o.fd).ToList();
        for (int i = 0; i < 10; i++)
        {
            this.dgGAtt.Rows.Add(gslist [0].name, gslist
[0].team, gslist [0].fd;
        }
        gslist =
resFoals.OrderByDescending(o=>o.fd).ToList();
        for (int i = 0; i < 10; i++)
        {
            this.dgFoals.Rows.Add(gslist [0].name, gslist
[0].team, gslist [0].fd;
        }
        gslist =
resPasses.OrderByDescending(o=>o.fd).ToList();
        for (int i = 0; i < 10; i++)
        {

```

```

        this.dgPasses.Rows.Add(gslist [0].name, gslist
[0].team, gslist [0].fd;
    }
    gslist =
resPick.OrderByDescending(o=>o.fd).ToList();
    for (int i = 0; i < 10; i++)
    {
        this.dgPick.Rows.Add(gslist [0].name, gslist
[0].team, gslist [0].fd;
    }
    gslist =
resMins.OrderByDescending(o=>o.fd).ToList();
    for (int i = 0; i < 10; i++)
    {
        this.dgMins.Rows.Add(gslist [0].name, gslist
[0].team, gslist [0].fd;
    }
    gslist =
resPens.OrderByDescending(o=>o.fd).ToList();
    for (int i = 0; i < 10; i++)
    {
        this.dgPens.Rows.Add(gslist [0].name, gslist
[0].team, gslist [0].fd;
    }
    gslist =
resPenSav.OrderByDescending(o=>o.fd).ToList();
    for (int i = 0; i < 10; i++)
    {
        this.dgPenSav.Rows.Add(gslist [0].name, gslist
[0].team, gslist [0].fd;
    }
    gslist =
resSaves.OrderByDescending(o=>o.fd).ToList();
    for (int i = 0; i < 10; i++)
    {
        this.dgSaves.Rows.Add(gslist [0].name, gslist
[0].team, gslist [0].fd;
    }
    gslist =
resScores.OrderByDescending(o=>o.fd).ToList();
    for (int i = 0; i < 10; i++)
    {
        this.dgScores.Rows.Add(gslist [0].name, gslist
[0].team, gslist [0].fd;
    }
    msdr.Close();

```

```

        conn.Exit();
    }

    private void MyTeamBtnClick(object sender, EventArgs
e)
    {
        MyTeamFrm mt = new MyTeamFrm ();
        mt.Show();
        this.Close();
    }

    private void TournamentsBtnClick(object sender,
EventArgs e)
    {
        TournamentsFrm tf = new TournamentsFrm ();
        tf.Show();
        this.Close();
    }

    private void TeamsBtnClick(object sender, EventArgs
e)
    {
        TeamsFrm tf = new TeamsFrm ();
        tf.Show();
        this.Close();
    }

    private void ProfileBtnClick(object sender, EventArgs
e)
    {
        ProfileFrm pf = new ProfileFrm ();
        pf.Show();
        this.Close();
    }

    private void ProfileBtnClick(object sender, EventArgs
e)
    {
        FootballerStatisticsFrm
pf = new FootballerStatisticsFrm
    ();
        pf.Show();
        this.Close();
    }

```

```
private void BestTeamBtnClick(object sender,  
EventArgs e)  
{  
    BestTeamFrm pf = new BestTeamFrm ();  
    pf.Show();  
    this.Close();  
}  
  
}
```

ДОДАТОК В
Слайди презентації

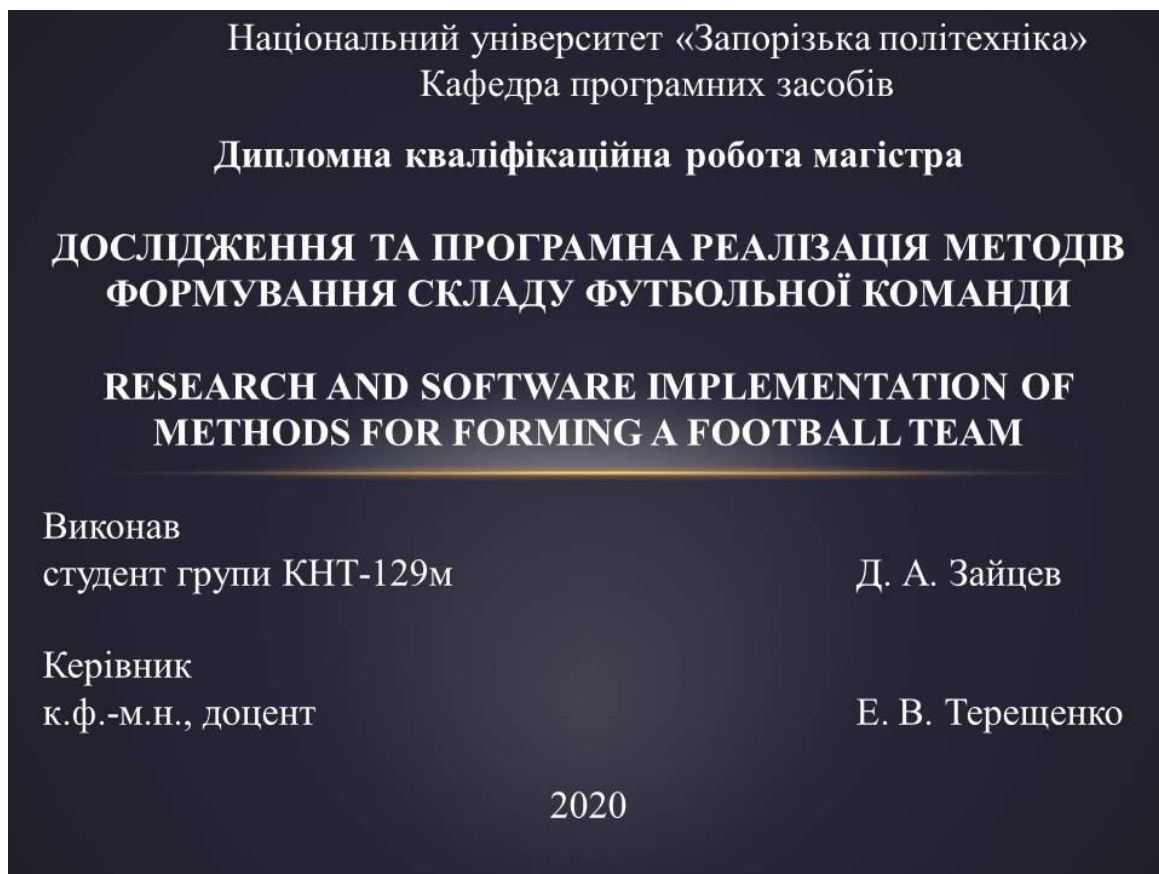


Рисунок В.1 – Слайд 1

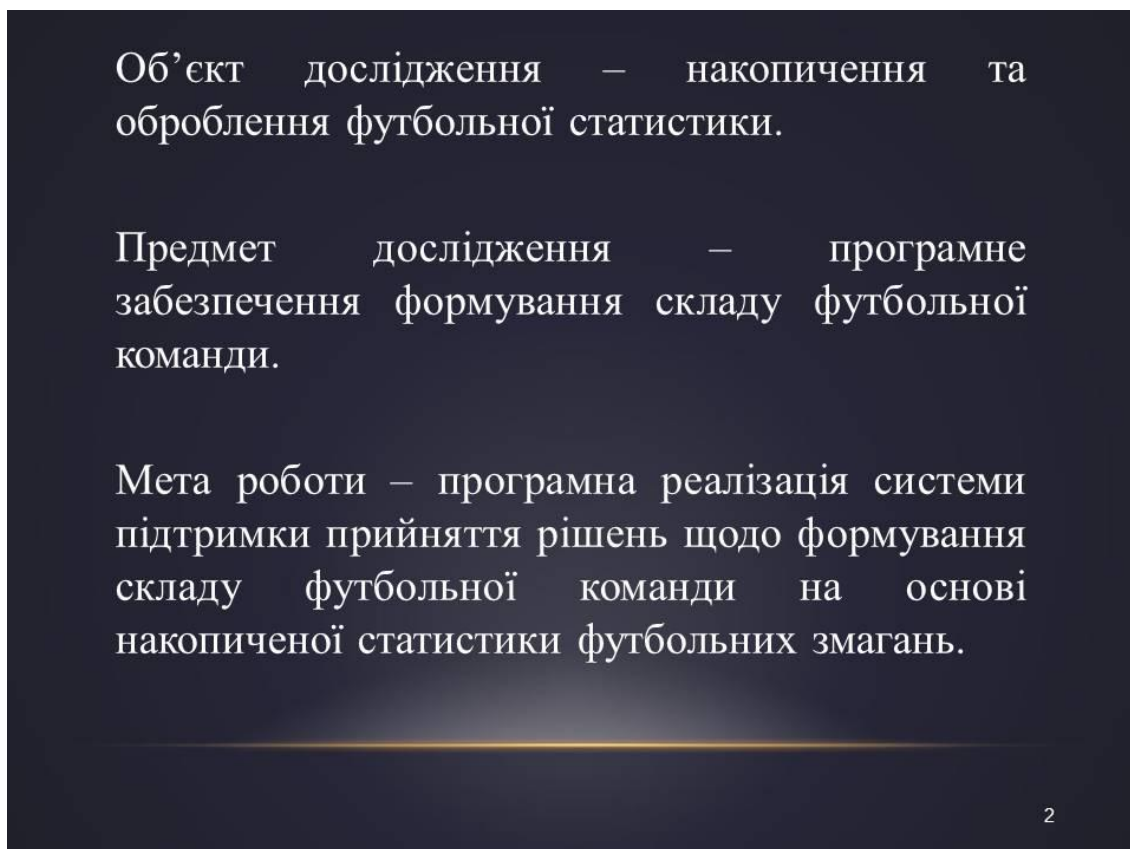


Рисунок В.2 – Слайд 2

Завдання роботи

- ✓ розроблення методу формування складу футбольної команди;
- ✓ проєктування програмного забезпечення;
- ✓ програмна реалізація та тестування програмного забезпечення.

3

Рисунок В.3 – Слайд 3

Оцінка виступу футболістів на основі очок

Категорія	Дія	Кількість очок
Всі гравці	Поява на полі	1
	60 хвилин, проведених на полі	2
	Гол з дальньої відстані (з-за меж штрафного майданчика)	1
	Результативна передача	3
	Кожні 3 відбори	1
	Зароблення пенальті для команди	2
	Зароблення пенальті у власні ворота	-1
	Незабитий пенальті	-2
	Жовта картка	-1
	Червона картка	-3
	Автогол	-2
	Гол	6
	Відбитий пенальті	5
	Матч без пропущених голів (відіграно не менше 60 хвилин)	4
	Кожні три порятунки	1
	Кожні два пропущені голи	-1

Категорія	Дія	Кількість очок
Захисник	Гол	6
	Матч без пропущених голів (відіграно не менше 60 хвилин)	4
	Кожні два пропущені голи	-1
Півзахисник	Гол	5
	Матч без пропущених голів (відіграно не менше 60 хвилин)	1
Нападник	Гол	4

4

Рисунок В.4 – Слайд 4

Задача формування складу футбольної команди

$$\sum_{j=1}^n s_j(a_{j-1}, d_j) \rightarrow \max,$$

$$n = n_g + n_d + n_m + n_f,$$

$$P_g + P_d + P_m + P_f \leq P,$$

$$P_g = \sum_{j=1}^{n_g} p_{g_j},$$

$$P_d = \sum_{j=1}^{n_d} p_{d_j},$$

$$P_m = \sum_{j=1}^{n_m} p_{m_j},$$

$$P_f = \sum_{j=1}^{n_f} p_{f_j},$$

$$d_j \in F,$$

де a_{j-1} – стан команди, якого було досягнуто на попередньому, $j - 1$, кроці;

d_j – рішення про вибір гравця в команду, що приймається на j -му кроці і виконується з множини F – множини всіх гравців, доступних на ринку;

$s_j(a_{j-1}, d_j)$ – сумарна оцінована (прогнозована) вартість очок, яку набере команда, сформована на j -му кроці;

n, n_g, n_d, n_m, n_f – кількість гравців, воротарів, захисників, півзахисників, нападників у команді;

P_g – сумарна вартість воротарів, за яку їх було придбано в команду;

P_d – сумарна вартість захисників, за яку їх було придбано в команду;

P_m – сумарна вартість півзахисників, за яку їх було придбано в команду;

P_f – сумарна вартість нападників, за яку їх було придбано в команду;

P – максимальний бюджет команди, тобто сумарна вартість всіх гравців команди;

5

Рисунок В.5 – Слайд 5

Порівняння архітектур нейронних мереж для розв'язання задач прогнозування

Критерій	Повнозв'язні нейронні мережі	Рекурентні нейронні мережі
Можливість прогнозування часового ряду	+	+
Врахування взаємозалежностей між значеннями	–	+
Область застосування	Універсальна	Прогнозування фінансових рядів, аналіз і генерація тексту

6

Рисунок В.6 – Слайд 6

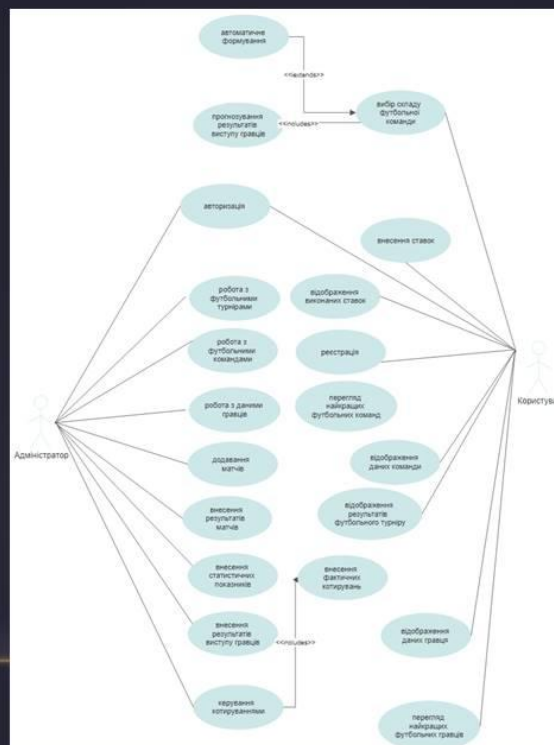
Метод формування складу футбольної команди

- На першому етапі методу має бути зібрано набір даних, що характеризують виступ кожного гравця, що буде брати участь у турнірі, на попередніх турнірах, і формується вибірка даних.
- На другому етапі виконується створення моделі нейронної мережі. Модель створюється на основі простої рекурентної нейронної мережі. Було використано структуру на основі вхідного, двох прихованих рекурентних та вихідного шарів.
- На третьому етапі реалізується навчання рекурентної нейронної мережі на сформованій на першому етапі вибірці.
- На четвертому етапі забезпечується прогнозування виступу кожного гравця за k попередніми значеннями для $k+1$ невідомого періоду часу.
- На п'ятому етапі реалізується метод динамічного програмування для виконання оптимального вибору. Оптимальний вибір є результатом роботи методу формування складу футбольної команди і рекомендується менеджеру футбольної команди для прийняття рішення.

7

Рисунок В.7 – Слайд 7

Сценарії використання програми



8

Рисунок В.8 – Слайд 8

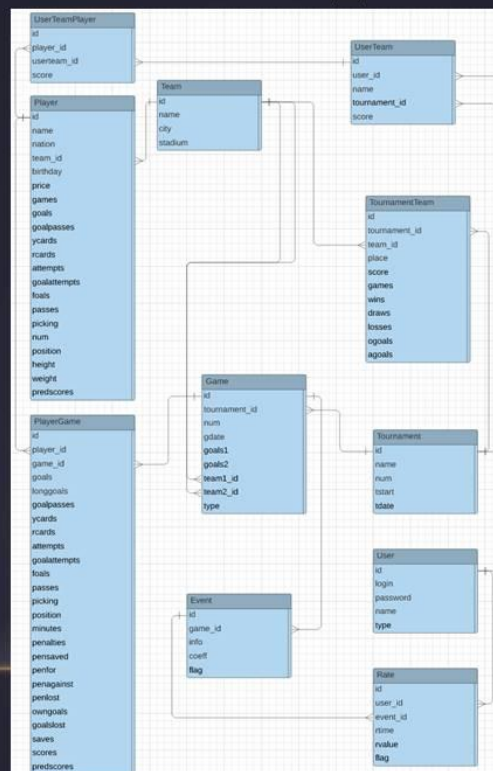
Порівняння мов програмування

Критерій	C#	C++
Парадигми програмування	Мультипарадигмальна мова з об'єктно-орієнтованим програмуванням на основі низькорівневих абстракцій	Мультипарадигмальна мова з об'єктно-орієнтованим програмуванням на основі високорівневих абстракцій
Залежність від платформи	Від платформи не залежить	Від платформи не залежить, але основний напрямок – Windows
Підтримка типів даних	Статична, обмежена	Статична/динамічна, більша кількість
Область застосування	Застосунки для настільних систем, вебзастосунки	Ігри, драйвери для пристроїв, серверні застосунки

9

Рисунок В.9 – Слайд 9

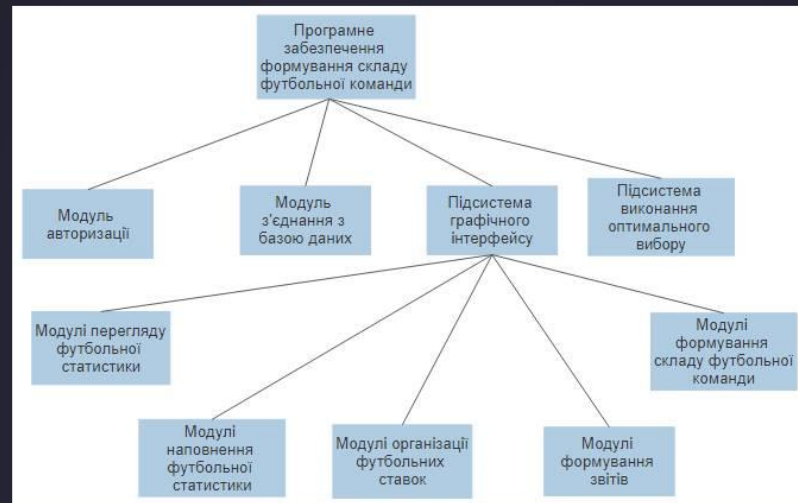
Схема бази даних



10

Рисунок В.10 – Слайд 10

Схема структури програми



11

Рисунок В.11 – Слайд 11

Форми адміністратора

Турніри

Назад Вийти

Новий Існуючий

Назва: Іспанська Ла Ліга

Кількість команд: 20

Початок: 01.09.2021

Завершення: 15.05.2022

Створити

Сезон: 2020/2021

Створені:

- Англійська Прем'єр-ліга
- Іспанська Ла Ліга
- Німецька Бундесліга
- Італійська Серія А
- Французька Ліга 1

Нова гра

Назад Вийти

Сезон: 2020/2021

Турнір: Англійська Прем'єр-ліга

Тур: 12

Перша команда: Манчестер Юнайтед

Друга команда: Манчестер Сіті

Дата: грудень 2020 р.

Пн Вт Ср Чт Пт Сб Нд

49 30 1 2 3 4 5 6

50 7 8 9 10 11 12 13

51 14 15 16 17 18 19 20

52 21 22 23 24 25 26 27

53 28 29 30 31 1 2 3

2 4 5 6 7 8 9 10

Сьогодні: 04.12.2020

Нова ставка:

Коефіцієнт:

Додати

Ставки:

Перемога першої команди	2.1
Перемога другої команди	1.9
Нічия	1.4

Створити

12

Рисунок В.12 – Слайд 12

Футбольна статистика

Моя команда

Турніри

Команди

Найкращі

Профіль

Вийти

Інформація про гравців

Сезон: 2020/2021

Турнір: Англійська Прем'єр-ліга

Команда: Тоттенгем

Гравець: Гаррі Кейн

Національність: англійський

Народження: 28.07.1993

Ігровий номер: 10

Висота: 11

Позиція: нападник

Ріст: 188

Вага: 75

Сума: Матч

Проведені ігри	11	Відбори	8
Голи	8	Жовті картки	1
Голи в передачах	10	Червоні картки	0
Удари по воротам	42	Порушення	15
Голи в ударах	23	Передачі	268

Профіль

Вийти

Таблиця

Сезон: 2020/2021

Турнір: Іспанська Ла Ліга

Місце	Команда	Ігри	В	Н	П	Матч	Очки
1	Атлетіко	10	8	2	0	21-2	26
2	Реал Сосьєдад	11	7	3	1	22-5	24
3	Вільярреал	12	5	6	1	15-11	21
4	Реал Мадрид	11	6	2	3	17-12	20
5	Кадіс	10	5	3	4	11-13	18
6	Севілья	10	5	1	4	12-9	16
7	Гранада	11	4	3	4	14-20	15
8	Бетіс	12	5	0	7	14-23	15
9	Барселона	10	4	2	4	20-11	14
10	Ельче	10	3	5	2	9-10	14
11	Атлетік	11	4	1	6	12-12	13
12	Ейбар	11	3	4	4	8-9	13
13	Алавес	11	3	4	4	11-13	13
14	Гетіфе	11	3	4	4	9-13	13
15	Сельта	12	3	4	5	13-20	13
16	Валенсія	11	3	3	5	17-17	12
17	Леванте	11	2	5	4	13-15	11
18	Осасуна	11	3	2	6	8-15	11
19	Вальядолід	12	2	4	6	11-18	10
20	Уеска	12	0	8	4	11-20	8

13

Рисунок В.13 – Слайд 13

Автоматизоване формування складу футбольної команди

Рейтинг

Гравці

Турніри

Команди

Найкращі

Профіль

Вийти

Формування команди

Турнір: Ліга Чемпіонів

Схема: 4-3-3

Команда: Аталанта

Бюджет: 94.5/100

Гравець: 6.5

Крістіан Ромеро

Команда: Манчестер Юнайтед

Гравець: 12.1

Маркус Рашфорд

Команда: Брюгге

Гравець: 6

Шарль Де Кетеларе

Команда: Істанбул

Гравець: 8.7

Ірфан Кахведжі

Команда: РБ Зальцбург

Гравець: 9.5

Мергін Беріша

Команда: Порту

Гравець: 6.4

Заїду Занусі

Команда: Челсі

Гравець: 6.9

Едуар Менді

Команда: Борусія Дортмунд

Гравець: 8

Рафаель Гереро

Команда: РБ Лейпциг

Гравець: 7.2

Анхеліно

Автоматично

Зберегти

14

Рисунок В.14 – Слайд 14

Експериментальне дослідження прогнозування кількості очок, набраних гравцями

Модель	Середньоквадратична помилка
Регресійна модель	0.1794
Повнозв'язна нейронна мережа	0.0536
Проста рекурентна нейронна мережа	0.0317

15

Рисунок В.15 – Слайд 15

Висновки

Наукова новизна роботи полягає в запропонованому методі формування складу футбольної команди на основі оптимізації сумарної оцінки виступу футбольної команди, виконуючи оптимізацію на основі методу динамічного програмування, а прогнозування індивідуальних виступів на основі рекурентних нейронних мереж, що дозволяє в підсумку виконувати автоматизоване прийняття рішень щодо вибору складу футбольної команди з обмеженням щодо вартості.

Для вирішення практичної проблеми було виконано моделювання сценаріїв роботи адміністратора та користувача з програмою, обрано мову C# для реалізації програмного забезпечення, реалізовано визначення структури бази даних за допомогою використання MySQL в якості інструменту роботи з бази даними, визначено структуру програми, описано процес реалізації програми. Розроблено програмне забезпечення у вигляді настільного застосунку.

Розроблене програмне забезпечення може використовуватися для керування футбольними командами, отримання доступу до футбольної статистики та організації віртуальних футбольних турнірів.

Усі завдання дипломної кваліфікаційної роботи магістра виконано повністю.

16

Рисунок В.16 – Слайд 16