

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕКОНОМІКИ І МЕНЕДЖМЕНТУ**

Кафедра кібернетики та інформаційних технологій

Кваліфікаційна робота

Освітній ступінь – «Бакалавр»

**на тему: Створення навчально інформаційної системи-симулятора
діяльності фірми**

Виконала:

студентка спеціальності

126 «інформаційні системи та технології»

ОП «Інформаційні системи та технології»

Хіманова Анастасія Олександрівна

Керівник:

В'юненко Олександр Борисович

доцент кафедри кібернетики та інформатики

Рецензент:

Ободяк Віктор Карнелійович

к.т.н., доцент кафедри кібербезпеки
Сумського державного університету

Суми – 2023

СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

Факультет	Економіки і менеджменту
Кафедра	Кібернетики та інформатики
Освітній ступінь	«Бакалавр»
Спеціальність	126 «Інформаційні системи та технології» ОП «Інформаційні системи та технології»

Затверджую:
Завідувач кафедри Світлана АГАДЖАНОВА

«_____» «_____» 2023____ р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ *Хімановій Анастасії Олександрівні*

1. Тема роботи: Створення навчально інформаційної системи-симулятора діяльності фірми

2. База практичного дослідження: Сумський НАУ.

керівник роботи В'юненко Олександр Борисович, к.е.н., доцент

затверджена наказом по університету від «_____» «_____» 2023 р. №_____

3. Строк подання студентом закінченого проекту (роботи)

«23» червня 2023 року

4. Вихідні дані до проекту (роботи):

Результати аналітичних досліджень зарубіжних та вітчизняних вчених, статистична звітність,, Інтернет-джерела.

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

- дослідити основні вимоги щодо створення систем-симуляторів
- проаналізувати особливості роботи інформаційних систем симуляторів
- проаналізувати цільову аудиторію використання системи
- розробити систему симулятор виконання проектів
- розробити систему симулятор фінансових операцій

6. Дата видачі завдання:

«9» жовтня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту(роботи)	Строк виконання етапів проекту	Примітка
1	Визначення об'єкту, предмету дослідження, формулювання мети та задач кваліфікаційної роботи, складання плану	вересень 2022 р.	виконано
2	Підбір та вивчення літературних джерел	жовтень - листопад 2022 р.	виконано
3	Узагальнення теоретичного матеріалу з обраної теми дослідження та представлення чорнового варіанту першого розділу кваліфікаційної роботи	грудень 2023 р.	виконано
4	Збір та обробка фактичного матеріалу, узагальнення аналізу застосування досліджуваного питання в установі (переддипломна практика)	січень-лютий 2023 р.	виконано
5	Оформлення теоретичної частини кваліфікаційної роботи, узагальнення аналітичної частини, захист звіту по переддипломній практиці	березень-квітень 2023 р.	виконано
6	Підготовка висновків, пропозицій та узгодження з керівником	травень 2023 р.	виконано
7	Підготовка кінцевого варіанту кваліфікаційної роботи	до 6 червня 2023 р.	виконано
8	Опрацювання зауважень керівника, перевірка на автентичність	до 9 червня 2023 р.	виконано
9	Перевірка з академічної доброчесності кваліфікаційної роботи	9 червня 2023 р.	виконано
10	Представлення кваліфікаційної роботи на кафедрі, деканат	12 червня 2023 р.	виконано
11	Підготовка доповіді, презентації до кваліфікаційної роботи та її попередній захист	червень 2023 р.	виконано
12	Захист кваліфікаційної роботи	28 червня 2023р.	виконано

Студент

(підпис)

Анастасія ХІМАНОВА

Керівник роботи

(підпис)

Надія ПАСЬКО

Перевірку на автентичність проведено

Надія БАРАНИК

Дипломна робота допущена до захисту

(підпис)

Маргарита ЛИШЕНКО

АНОТАЦІЯ

Хіманова А.О. Створення навчально інформаційної системи-симулятора діяльності фірми

Кваліфікаційна робота зі спеціальності 126 «Інформаційні системи та технології» ОП Інформаційні системи та технології, СНАУ, Суми-2023р. – Рукопис.

Дана кваліфікаційна робота присвячена створенню навчально-інформаційної системи-симулятора діяльності фірми. Основною метою є надання користувачам можливості взаємодіяти з віртуальною фірмою та навчитися принципам та процесам, пов'язаним з управлінням бізнесом.

Для досягнення цієї мети розроблено навчально-інформаційну систему-симулятор, яка використовує сучасні технології веб-розробки. Система надає користувачам можливість взаємодіяти з віртуальною фірмою, що моделює реальність бізнес-середовища. Вона пропонує користувачам виконувати різні ролі в організації, приймати управлінські рішення на основі доступних даних, а також ознайомитися з куплею та продажами акцій фірм

Кваліфікаційна робота реалізує навчально інформаційну систему, яка дозволяє користувачам отримати практичні навички управління фірмою та розуміння впливу фінансових рішень на кінцеві результати. Система дозволяє відтворити реалістичну ситуацію бізнес-середовища та забезпечує ефективну взаємодію користувача з основними аспектами діяльності фірми.

Ключові слова: система-симулятор, управління проектами

SUMMARY

Khimanova A.O. Creation of an educational information system-simulator of the company's activity

Qualification work on specialty 126 "Information systems and technologies" OP Information systems and technologies, SNAU, Sumy-2023. - Manuscript.

This qualification work is dedicated to the creation of an educational and information system-simulator of the company's activities. The main goal is to give users the opportunity to interact with a virtual firm and learn the principles and processes related to business management.

To achieve this goal, an educational and information system-simulator was developed, which uses modern web development technologies. The system provides users with the opportunity to interact with a virtual firm that simulates the reality of the business environment. It offers users to perform various roles in the organization, make management decisions based on available data, and also familiarize themselves with the work of the stock exchange.

The qualification work implements an educational information system that allows users to gain practical skills in managing a firm and understanding the impact of financial decisions on final results. The system allows you to reproduce a realistic situation of the business environment and ensures effective interaction of the user with the main aspects of the company's activities.

Key words: simulator system, project management, stock exchange

ЗМІСТ

ВСТУП.....	6
РОЗДІЛ 1. ПРЕДМЕТНА ОБЛАСТЬ	7
1.1 Визначення поняття комп'ютерної гри	7
1.2 Комп'ютерні бізнес-симулятори	9
1.3 Постановка задачі.....	12
РОЗДІЛ 2. МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ.....	15
2.1 Вибір середовища проектування та мови програмування.....	15
2.2 Проектне управління	19
2.3 Проектування бази даних.....	23
РОЗДІЛ 3. РОЗРОБКА ІГРОВОГО СИМУЛЯТОРУ.....	26
3.1 Архітектура симулятора.....	26
3.2 Розробка симулятора виконання проектів	28
3.3 Розробка симулятора фінансових операцій	39
ВИСНОВОК.....	45
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	46
ДОДАТКИ.....	47
ДОДАТОК А. Технічне завдання	48
ДОДАТОК Б. Лістинг програмного коду	57
ДОДАТОК В. Тези	110

ВСТУП

Актуальність використання навчальних симуляторів: сучасний світ комп'ютерних наук постійно розвивається, вимагаючи від фахівців у цій галузі широкого спектру знань та навичок. Одним із важливих аспектів є розуміння процесів та етапів роботи в сфері комп'ютерних проектів, а також вміння ефективно керувати ними.

Процес розробки програмного забезпечення й відповідні професійні компетентності майбутніх інженерів-програмістів належать до тих професійних компетентностей, що вкрай важко сформувати лише за допомогою лекційних занять. Практичні заняття і проекти, що досить часто доповнюють лекції, також обмежені їх академічною постановкою. Вважається, що використання ігрових симуляторів має великий потенціал у розширенні навчального досвіду поза межами лекцій і практичних занять, сприяє розумінню майбутніми інженерами програмістами необхідності формування професійних компетентностей і значно розширює кар'єрні перспективи майбутніх фахівців.

Зазначимо, що під навчальними симуляторами розуміються інтерактивні програми, які виходять за межі традиційного ринку відеоігор, що симулюють повністю або частково певні реальні процеси або системи й використовуються у навчальному процесі у випадках, коли реальна практика не є можливою або не є доступною. Отже, ігрові симулятори є надзвичайно потужним освітнім інструментом, який, що важливо, захоплює і мотивує студентів за допомогою веселого та цікавого ігрового досвіду.

Мета кваліфікаційної роботи: розробити навчально інформаційну систему-симулятор діяльності фірми.

Задачі:

- 1) Розібратися з поняттями ділова гра та інформаційні симулятори;
- 2) Обрати середовище програмування;
- 3) Розробити симулятор виконання проектів;
- 4) Розробити симулятор фінансових операцій.

РОЗДІЛ 1

ПРЕДМЕТНА ОБЛАСТЬ

1.1 Визначення поняття комп'ютерної гри

Комп'ютерна гра є одним з найпопулярніших форм електронних розваг та відпочинку, яка захоплює мільйони людей по всьому світу. Вона представляє собою програмне забезпечення, розроблене для відтворення імітаційного середовища або віртуального світу, в якому гравець може взаємодіяти з об'єктами та персонажами, керувати їх діями та вирішувати завдання заздалегідь встановленої гри.

Комп'ютерні ігри можуть мати різні жанри та варіації, включаючи стратегічні ігри, рольові ігри, екшн-ігри, головоломки, імітаційні ігри та багато інших. Кожен жанр має свої особливості, правила та механіку гри, що визначають спосіб взаємодії гравця з грою та її ціль.

Основні складові комп'ютерної гри включають:

- Графіка: візуальний аспект гри, який включає зображення, анімацію та ефекти; візуальний аспект гри може бути реалістичним, карикатурним або стилізованим, залежно від задуму розробників.
- Звук: аудіо компонент гри, який включає фонову музику, звукові ефекти, діалоги персонажів та інші звукові елементи; якісний звук може покращити імерсивність гри та створити атмосферу.
- Геймплей: це механіка гри, яка описує правила, умови, обмеження та способи взаємодії гравця з грою; геймплей визначає мету гри, способи досягнення цілей та вирішення завдань.
- Історія та сюжет: деякі комп'ютерні ігри мають заплановану історію або сюжет, який надає контекст та мотивацію гравцю; це може включати введення персонажів, відносини між ними та основні події, що розвиваються протягом гри.

– Взаємодія: комп'ютерна гра надає гравцю можливість взаємодіяти з грою через керування персонажем або об'єктами; це може бути здійснене за допомогою клавіатури, миші, геймпада або інших пристроїв.

Комп'ютерні ігри можуть мати різноманітні жанри, які визначають їх основну тематику та стиль геймплею. Основні жанри комп'ютерних ігор включають:

– Екшн (Action): цей жанр характеризується швидким темпом гри, бойовими сценами та інтенсивними діями; гравець часто керує головним героєм, який здійснює боротьбу зі ворогами, виконує складні маневри та вирішує завдання в реальному часі.

– Рольові ігри (Role-playing Games, RPG): RPG ігри фокусуються на розвитку персонажа гравця, його взаємодії зі світом гри та виконанні завдань; вони можуть мати елементи фантазії, наукової фантастики або історичного фону; гравець зазвичай приймає роль персонажа зі своїми унікальними характеристиками, вміннями та можливостями.

– Стратегії (Strategy): стратегічні ігри зосереджені на плануванні, управлінні ресурсами та вирішенні стратегічних завдань; гравець має контролювати економіку, армію, технології та інші аспекти, щоб досягти перемоги; цей жанр може включати піджанри, такі як стратегії в реальному часі (Real-Time Strategy, RTS) та пошагові стратегії (Turn-Based Strategy).

– Головоломки (Puzzle): головоломкові ігри базуються на розв'язанні різних логічних задач і головоломок; вони вимагають від гравця аналітичних та творчих навичок для знаходження правильних рішень; цей жанр може включати головоломки на логіку, фізику, криптографію та багато інших.

– Пригоди (Adventure): пригодницькі ігри ставлять гравця в роль героя, який проводить розслідування, вирішує головоломки та досліджує віртуальний світ; цей жанр часто має насичений сюжет, діалоги з персонажами та варіативний розвиток подій.

– Симулятори (Simulation): симулятори намагаються точно відтворити реальні процеси або ситуації. Вони можуть включати авіасимулятори, автомобільні

симулятори, економічні симулятори, симулятори будівництва та багато інших; гравець має можливість керувати різними аспектами системи та спостерігати за їх впливом на результати.

Ці жанри не є вичерпним списком, а лише представляють основні категорії комп'ютерних ігор. Розвиток технологій і креативність розробників продовжує вводити нові жанри та комбінації.

1.2 Комп'ютерні бізнес симулятори

Бізнес-симулятори є ефективним інструментом для навчання діловим навичкам і забезпечують віртуальне середовище для прийняття рішень. Вони пропонують практичний підхід до навчання, дозволяючи студентам застосовувати теоретичні знання в практичних сценаріях. За словами Майкла, Чена та Алаві (2016), ці ігри широко використовуються в навчальних закладах і показали позитивні результати в покращенні розуміння студентами бізнес-процесів, стратегічного мислення та навичок вирішення проблем. Автори рекомендують найкращі практики для інтеграції бізнес-симуляційних ігор у навчальний план, щоб максимізувати їхні освітні переваги.

Комп'ютерні бізнес-симулятори є потужними інструментами для моделювання та віртуального відтворення реальних бізнес-середовищ. Вони надають можливість гравцям відчувати себе у ролі підприємців, менеджерів або стратегічних лідерів і дослідити різні аспекти бізнесу, такі як управління, прийняття рішень, маркетинг, фінанси та взаємодія з конкурентами на віртуальному ринку. У цьому розділі ми детальніше розглянемо комп'ютерні бізнес-симулятори, їхню функціональність та значення у сучасному бізнес-навчанні та дослідженнях [16].

Існує різноманітність комп'ютерних бізнес-симуляторів, які варіюються за тематикою, рівнем складності та цілями, що ставляться перед гравцями. Деякі з них сфокусовані на певних аспектах бізнесу, таких як управління ресурсами, логістика або стратегічне планування, тоді як інші намагаються відтворити повний спектр

бізнес-операцій. Деякі приклади типів комп'ютерних бізнес-симуляторів включають:

- Фінансові симулятори: ці симулятори фокусуються на аспектах фінансового управління, таких як управління активами та пасивами, інвестиції, фінансовий аналіз та ризик-менеджмент; вони дозволяють гравцям відтворити ситуації з реального фінансового світу та приймати рішення щодо інвестиційного портфеля, фінансових стратегій та управління ризиками.

- Управління проектами: цей тип симуляторів дозволяє гравцям відчувати себе у ролі проектного менеджера та керувати проектами різної складності; вони включають в себе етапи планування, виконання, контролю та закриття проекту, допомагаючи гравцям розвивати навички управління ресурсами, комунікації та прийняття рішень у проектному середовищі.

- Стратегічні симулятори: ці симулятори спрямовані на розвиток стратегічного мислення та управління; вони дозволяють гравцям взяти на себе роль керівників організацій або лідерів цивілізацій, де вони займаються прийняттям стратегічних рішень, розвитком бізнес-планів, дослідженнями ринку та конкуренцією.

- Менеджмент ресурсів: ці симулятори фокусуються на управлінні різними видами ресурсів, такими як людські ресурси, матеріали, енергія тощо; гравці беруть на себе роль керівників компаній або фабрик і приймають рішення щодо ефективного використання ресурсів, оптимізації виробництва та досягнення позитивних показників продуктивності.

Комп'ютерні бізнес-симулятори відіграють важливу роль у сучасному бізнес-навчанні. Вони допомагають студентам отримати практичні навички та дослідити різні аспекти бізнесу, що доповнює теоретичне навчання. Основні переваги використання комп'ютерних бізнес-симуляторів у бізнес-навчанні включають:

- Активне навчання: симулятори створюють можливість для активного навчання, де студенти можуть самостійно вирішувати проблеми, приймати рішення та відчувати наслідки своїх дій; це сприяє кращому засвоєнню матеріалу та розвитку критичного мислення.

- Реалістичність: комп'ютерні бізнес-симулятори добре відтворюють реальні бізнес-ситуації, що дозволяє студентам отримати реалістичний досвід і зрозуміти складнощі та виклики, з якими стикаються бізнес-практики.

- Тестування стратегій: симулятори дозволяють студентам експериментувати з різними стратегіями та оцінювати їх ефективність у безрисковому середовищі; вони можуть спробувати нові ідеї, здійснювати зміни у виробничих процесах та вивчити наслідки своїх рішень.

В даний час бізнес симулятори використовуються в навчальному процесі інститутів підвищення кваліфікації, на науково-методичних конференціях та навчальному процесі, як у вищих, так і середніх спеціальних навчальних закладах, технічних та гуманітарних. Широке поширення спроб застосування ділових ігор має свої позитивні та негативні сторони і відповідно своїх прихильників та противників. З'явилися дві протилежні тенденції її осмислення. Позитивна підтверджує можливості ділових ігор як інструменту формування особистості спеціаліста та активізації навчального процесу. Негативна пов'язана з недостатньо глибоким розумінням сутності бізнес симулятора. Слід зазначити, що у науковій літературі відсутня загальноприйнята концепція бізнес симулятора.

Бізнес симулятори у своїй основі мають пошук оптимальних рішень конкретної проблеми чи завдання, що виникають у практичній діяльності. Ми вважаємо, що це дуже ефективний та корисний вид діяльності для учнів, студентів та працівників в даний час.

Бізнес симулятори можуть застосовуватися в навчанні, діагностиці індивідуальних особливостей їх учасників, організації процесу прийняття рішень, в дослідницьких цілях. Можна звернути увагу на те, що даний метод синтезує переваги експериментального, аналітичного та експертного методів.

Розглянемо переваги бізнес симуляторів перед традиційними методами навчання:

- Цілі симулятора більшою мірою узгоджуються з практичними потребами студентів. Дана форма організації навчального процесу знімає суперечність між абстрактним характером навчального предмета та реальним характером

професійної діяльності, системним характером використовуваних знань та їх приналежності до різних дисциплін.

- Метод дозволяє поєднати широке охоплення проблем та глибину їх осмислення.

- Ігрова форма відповідає логіці діяльності, включає момент соціальної взаємодії, готує до професійного спілкування.

- Ігровий компонент сприяє більшій залученості учнів.

- Бізнес симулятор насичений зворотним зв'язком, причому більш змістовним в порівнянні з застосовуваним у традиційних методах.

- У симуляторі формуються настанови професійної діяльності, легше долаються стереотипи, коригується самооцінка.

- Традиційні методи передбачають домінування інтелектуальної сфери, у грі проявляється вся особистість.

- Даним методом стимулює включення рефлексивних процесів, представляє можливість інтерпретації, осмислення отриманих результатів [3].

Бізнес симулятори дозволяють збільшити масштаб охоплення дійсності, наочно наслідки прийнятих рішень, дають можливість перевірити альтернативні рішення. Інформація, якою користується людина насправді, неповна, неточна. У симуляторі йому надається хоч і неповна, але точна інформація, що підвищує довіру до отриманих результатів та стимулює процес прийняття відповідальності.

1.3 Постановка задачі

Сучасні технології та інформаційні системи відіграють важливу роль у сучасному бізнесі. Однак, навчання студентів та молодих фахівців у сфері управління проектами, командної роботи та прийняття стратегічних рішень часто відбувається в теоретичному середовищі без практичного застосування знань. Це може призводити до недостатньої підготовки та неефективної роботи у реальних умовах.

Отже, виникає потреба у створенні практичного інструменту, що дозволить студентам і молодим фахівцям набути практичних навичок управління проектами та розвитку командної роботи. Відповідно до цього, ми обрали тему "Створення навчальної інформаційної системи-симулятора діяльності фірми" для нашого проекту.

Головною метою нашого проекту є розробка навчальної інформаційної системи-симулятора, яка забезпечить користувачам можливість практичного навчання та розвитку навичок управління проектами, командної роботи та стратегічного мислення. Ми плануємо створити інтерактивне віртуальне середовище, де користувачі зможуть отримати основні навички управління та фінансової грамотності

Для досягнення поставленої мети, ми визначили наступну структуру симулятору:

1. Головне меню:
 - Перехід на HTML-сторінку фінансового симулятора.
2. HTML-сторінка фінансового симулятора:
 - Можливість взяти позику в банку під відсоток.
 - Можливість вкладати кошти в акції.
 - Повернення до головного меню
3. Створення нового проекту:
 - Введення назви проекту.
 - Виведення суми, яку вам заплатять за цей проект.
4. Вибір програміста:
 - Вибір програміста з певною зарплатою.
 - Перехід до міні-гри.
5. Міні-гра для програміста:
 - Рахування часу, де 10 секунди відповідають одному дню.
 - Підрахунок витрат на програміста.
 - Запис даних до бази даних.

6. Вибір дизайнера:

- Вибір дизайнера з відповідною зарплатою.
- Перехід до міні-гри

7. Міні-гра для дизайнера:

- Запис днів та розрахунок зарплати.
- Запис даних до бази даних.

8. Вибір тестувальника:

- Вибір тестувальника.
- Перехід до міні-гри

9. Міні-гра для тестувальника:

- Запис днів та розрахунок зарплати.
- Запис даних до бази даних.

10. Форма результатів проекту:

- Виведення суми, яку вам заплатили за проект.
- Виведення суми, яку ви витратили на програміста, дизайнера та тестувальника разом.
- Виведення залишку коштів.
- Розрахунок залишку коштів шляхом віднімання витрат від отриманої суми за проект.
- Запис даних до бази даних.

Ця структура дозволить користувачу пройти через різні етапи проектування, взаємодіяти з різними співробітниками, грати міні-ігри та виводити результати проекту. Запис даних до бази даних дозволить зберігати інформацію про кожен етап та здійснювати аналіз після завершення проекту.

У цьому розділі було розглянуто постановку задачі для нашого проекту "Створення навчальної інформаційної системи-симулятора діяльності фірми". Визначені цілі та завдання проекту, а також обґрунтовано вибір даної теми. У наступних розділах ми розглянемо детальніше реалізацію проекту та його функціональні можливості.

РОЗДІЛ 2

МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ

2.1 Вибір середовища проектування та мови програмування

Порівняємо середовища програмування Geany, Microsoft Visual Studio, SharpDevelop та MonoDelop, порівняння наведено в таблиці 2.1.

Таблиця 2.1 - Порівняння засобів програмування

Середовище розробки	Розробник	Платформа	Ліцензія
Geany	Team	UNIX/Windows	GPL
Microsoft Visual Studio	Microsoft	Windows	Закрита
MonoDevelop	Novell i Mono community	Cross-platform	GPL
SharpDevelop	ICSharCode Team	Windows	LGPL

Джерело: запропоновано автором

Ліцензія GPL надає користувачеві права копіювати, модифікувати та розповсюджувати (у тому числі на комерційній основі) програми (що за замовчуванням заборонено законом про авторські права), а також гарантує, що й користувачі всіх похідних програм отримають перераховані вище права[4].

Ліцензія LGPL дозволяє лінкувати з цією бібліотекою або програмою програми під будь-якою ліцензією, несумісною з GNU GPL, за умови, що така програма не є похідною від об'єкта, що розповсюджується під (L)GPL, окрім шляхом лінкування. Головна різниця між GPL і LGPL у тому, що остання дозволяє і таке лінкування з даним об'єктом інших, яке створює похідну від даної роботи, якщо ліцензія об'єктів злінкованих дозволяє «модифікації для внутрішнього використання споживачем і зворотну розробку для налагодження таких модифікацій». Тобто. LGPL, на відміну від GPL, дозволяє зв'язування бібліотеки з будь-якою програмою, не обов'язково вільною.

Закрите (пропрієтарне) програмне забезпечення (англ. proprietary software) - програмне забезпечення, яке є приватною власністю авторів або правовласників і не відповідає критеріям вільного ПЗ (наявності відкритого програмного коду

недостатньо). Правовласник пропрієтарного ПЗ зберігає у себе монополію з його використання, копіювання і модифікацію, повністю чи суттєвих моментах. Зазвичай пропрієтарним називають будь-яке невільне ПЗ, включаючи напіввільне.

Geany – вільне середовище розробки програмного забезпечення, написане з використанням бібліотеки GTK2. Доступна для таких операційних систем: BSD, Linux, Mac OS X, Solaris та Windows. Geany поширюється згідно з GNU General Public License. Geany не включає компілятор. Натомість використовується GNU Compiler Collection (або будь-який інший компілятор) для створення виконуваного коду.

Microsoft Visual Studio - лінійка продуктів корпорації Майкрософт, що включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Дані продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, у тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-програми, веб-служби як в рідному, так і в керованому кодах для всіх платформ, підтримуваних Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight .

MonoDevelop – вільне середовище розробки, призначене для створення додатків C#, Java, Boo, Nemerle, Visual Basic .NET, Vala, CIL, C та C++. Також планується підтримка Oxygene Embarcadero Technologies. Спочатку це був порт SharpDevelop на Mono/GTK+, але з того часу проект далеко втік від свого початкового стану. MonoDevelop є частиною проекту Mono.

SharpDevelop – вільне середовище розробки для C#, Visual Basic .NET, Boo, IronPython, IronRuby, F#, C++. Зазвичай використовується тими, хто хоче використовувати Visual Studio .NET. Існує також форк на Mono/Gtk+ - MonoDevelop. SharpDevelop 2.0 надає інтегрований налагоджувач, який використовує власні бібліотеки та взаємодіє з виконуючим середовищем .NET через COM Interop. Хоча SharpDevelop 2.0 (як і VS2005) використовує файли проекту у форматі MSBuild, він може використовувати компілятори від .NET Framework 1.0 і 1.1, а також від Mono.

Для розробки необхідно активно використовувати усі засоби мови програмування. Однак середовище MonoDevelop використовує власний компілятор, який не повністю підтримує мову C# через те, що є вільною мультиплатформною розробкою, незалежною від творців мови. Хоча вона забезпечує мультиплатформенність, але неможливо передбачити поведінку мови в нових версіях. А однією з ключових складових проекту є його стійкість до відмов і стабільність і в той же час мультиплатформенність не потрібна (користувачів ІС на Linux зникаюче мало). Тому це середовище не підходить для розробки цього проекту.

SharpDevelop та Geany не мають власних компіляторів. Тому для розробки з використанням цих середовищ все одно доведеться використовувати пропрієтарне програмне забезпечення, що робить їх використання виправданим лише в деяких випадках. Наприклад, на низькопродуктивних комп'ютерах або при дуже обмеженому бюджеті проекту. Незважаючи на те, що вони можуть запускатися і працювати в ОС Linux, дані середовища розробки через відсутність власних компіляторів не зможуть створити мультиплатформну програму, і розробка все одно обмежиться операційними системами Windows.

Microsoft Visual Studio також не позбавлена недоліків. Основними є важковаговість, потребує досить великий обчислювальної потужності комп'ютера; платність; відсутність мультиплатформенності. Незважаючи на ці недоліки, Visual Studio залишається середовищем розробки більшості C# програмістів. Причиною цього є повна підтримка мови, розширені засоби розробки, документація, що енергійно розвивається, і саме середовище. Це середовище розробки будемо використовувати у проекті.

Порівняємо мови програмування такі, як Smalltalk, C++, C#, Java та Eiffel, порівняння наведене в таблиці 2.2.

Порівнявши дані мови програмування для нашого проекту ми обрали саме C#. Вона є однією з основних мов програмування, які підтримуються платформою .NET Framework, яка включає Windows Forms. C# є потужною, ефективною та досить легкою для вивчення мовою, що робить її відмінним вибором для створення

навчальних систем. Вона має багато вбудованих функцій та бібліотек для роботи з графікою, мережами, базами даних та іншими аспектами, необхідними для розробки нашої системи. C# і Windows Forms повністю інтегровані з .NET Framework, що дає нам доступ до багатьох потужних функцій і бібліотек для розробки настільних додатків.

Таблиця 2.2 – Порівняння мов програмування

Параметри	Smalltalk	C++	C#	Java	Eiffel
Статична типізація	+/-	+	+	+	+
Динамічна типізація	+	-	-	-	-
Явна типізація	+/-	+	+	+	+
Неявна типізація	+	-/+	+/-	-	-
Шаблони	+	+	-	-	+
Створення об'єктів в стеці	?	+	+	+	-/+
Некеровані покажчики	+	+	+	-	-
Ручне управління пам'яттю	+	+	-	-	-
Збір сміття	+	-	+	+	+
Цілі числа довільної довжини	+	-	-	+	?
Перезапуск функцій	+	+	+	+	-
Значення параметрів по замовчуванню	+/-	+	-	-	-
Локальні функції	-	-	+/-	+/-	-
Цикл foreach	+	+/-	+	+	-
Підтримка виключень	+	+	+	+	+
Контактне програмування	-	-	-	+/-	+
Множинне наслідування	-	+	-	-	+

Джерело: запропоновано автором

Ми можемо використовувати готові компоненти та класи для реалізації потрібного функціоналу, що зменшує час розробки та спрощує підтримку системи. скільки ми створюємо систему для Windows, використання Windows Forms та мови C# забезпечує нам максимальну сумісність та оптимізацію для цієї платформи. Ми можемо використовувати всі переваги, які надає ОС Windows, такі як доступ до системних ресурсів, інтеграцію зі звичайними інструментами та додатками, підтримку мультипоточності та інші. Велика спільнота розробників C# та .NET Framework забезпечує нас великою кількістю додаткових бібліотек, компонентів та документації. Це дозволяє нам швидко розширювати функціонал системи та знайти

відповіді на будь-які питання або проблеми, з якими можемо зіткнутися під час розробки.

Також дана мова є чудовою для початку вивчення, оскільки має керовані частини, які полегшать вам розуміння того, що сталося, коли ваші програми працюють неправильно[17].

Загалом, використання мови C# та платформи Windows Forms дозволить нам створити зручну, функціональну та ефективну навчально-інформаційну систему-симулятор для діяльності фірми, яка буде сумісна з операційною системою Windows та легко розширюватись у майбутньому.

2.2 Проектне управління

Проектне управління (Project Management) є важливою складовою процесу розробки програмного продукту або реалізації будь-якого проекту. Включаючи симулятори, ефективне проектне управління допомагає забезпечити успішне завершення проекту в рамках встановлених термінів, обсягу робіт та бюджету. Воно є системою методів, інструментів та практик, які застосовуються для планування, організації, виконання, контролю та закриття проекту з метою досягнення його мети та завдань. Воно забезпечує ефективне використання ресурсів, включаючи людські, фінансові, матеріальні та інформаційні, для досягнення бажаних результатів проекту[13].

Основні принципи проектного управління є важливою складовою успішної реалізації будь-якого проекту, включаючи проект навчальної інформаційної системи-симулятора діяльності фірми. Ці принципи допомагають забезпечити ефективне планування, виконання та контроль проекту, а також досягнення поставлених цілей в умовах обмежених ресурсів[15].

Як бачимо, функціональність сучасних додатків для проектного управління дозволяє користувачу слідкувати за часом виконання задач проекту. Більшість даних користувач вводить сам та на ньому лежить відповідальність за планування

задачами та ресурсами. Власне, користувач самостійно розподіляє співробітників і сучасні додатки не пропонують автоматично розподілити співробітників або запропонувати хоч якусь допомогу у розподілу виконавців задач.

Порівняльна характеристика засобів проектування наведена в таблиці 2.3

Таблиця 2.3 - Порівняння функціоналу існуючих рішень

Назва системи	Wrike	Asana	ProjectManager	Microsoft Project
Реєстрація та авторизація	+	+	+	-
Управління проектом командою	+	+	+	-
Внесення даних в таблицю	+	-	+	+
Створення підзадач	+	+	+	+
Створення пріоритетів	+	+	+	+
Статус задач	+	+	+	+
Хронологія діаграми Ганта	+	+	+	+
Відображення критичного шляху	+	-	+	+
Відслідковування зайнятості співробітників	+	-	+	+
Безкоштовна версія без обмежень	-	-	+	-

Джерело: запропоновано автором

Незважаючи на те, що складання плану є в деякому роді процесом творчим, все-таки, для власної зручності бажано дотримуватися деяких простих правил:

- Створюється головна сумарна завдання - проект.
- Створюється підзадача - віха старту.
- Відповідно до ієрархічної структури робіт створюються сумарні завдання вузли ієрархічної структури робіт.
- Кожне завдання, обов'язково повинна мати тривалість (від 0 і до 34978 днів), зв'язок (якщо немає попередника - попередником призначається Віха старту, у якій попередника немає, а якщо ні послідовника - послідовником призначається Віха фінішу), ресурси (матеріальні, трудові, витрати). Ресурсу може не бути, якщо є така технологічна потреба.
- В кінці проекту ставиться Віха фінішу[2].

Діаграми Ганта — це інструменти для планування великих проектів, які допомагають розділити їх на завдання й підзавдання та розподілити їх на шкалі часу. Кожне завдання вказується на одній стороні графіка й має горизонтальну лінію навпроти нього, яка вказує тривалість його виконання. Відображаючи подібні завдання, можна легко зрозуміти, як довго вони триватимуть і які завдання йтимуть одночасно.

Спочатку ми створили новий проект вписавши туди послідовність дій написання програми, це наведено на рисунку 2.1.

	Режим задачі	Назва задачі	Длительность	Начало	Окончание
1		Аналіз методів реалізації	4 днів	Пн 09.01.23	Чт 12.01.23
2		Вибір платформи	2 днів	Пн 09.01.23	Вт 10.01.23
3		Вибір мови програмування	2 днів	Ср 11.01.23	Чт 12.01.23
4		Створення функціоналу	40 днів	Пт 13.01.23	Чт 09.03.23
5		Створення головних форм	6 днів	Пт 13.01.23	Пт 20.01.23
6		Написання коду	34 днів	Пн 23.01.23	Чт 09.03.23
7		Створення додаткових форм	25 днів	Пт 10.03.23	Чт 13.04.23
8		Налаштування додаткових	18 днів	Пт 10.03.23	Вт 04.04.23
9		Прописування можливих п	7 днів	Ср 05.04.23	Чт 13.04.23
10		Налаштування дизайну	16 днів	Пт 14.04.23	Пт 05.05.23
11		Оформлення форм, кнопок	13 днів	Пт 14.04.23	Вт 02.05.23
12		Додавання ресурсів	3 днів	Ср 03.05.23	Пт 05.05.23
13		Тестування програми	2 днів	Пн 08.05.23	Вт 09.05.23
14		Перевірка функціоналу та	2 днів	Пн 08.05.23	Вт 09.05.23
15		Допрацювання	4 днів	Ср 10.05.23	Пн 15.05.23
16		Допрацювання 1	2 днів	Ср 10.05.23	Чт 11.05.23
17		Допрацювання 2	2 днів	Пт 12.05.23	Пн 15.05.23

Рисунок 2.1 - Етапи виконання проекту

Джерело: запропоновано автором

Також додаток дає змогу розподіляти ресурси та встановлювати погодинну заробітну плату, за рахунок чого ми можемо вирахувати місячну зарплату кожного співробітника, це наведено на рисунку 2.2.

Прописавши всі кроки, ми встановили між ними зв'язки, після цього вказали дату початку проекту і вписавши приблизні терміни їх виконання, перед нами з'явилась дата приблизного завершення та графік зі зв'язками між етапами проекту

На етапі створення діаграми деяка частина проекту вже була виконана, тому ми вказали це в відсотках на кожний етап, гарно видно процес на діаграмі Ганта з відслідковуванням, наведена на рисунку 2.3.

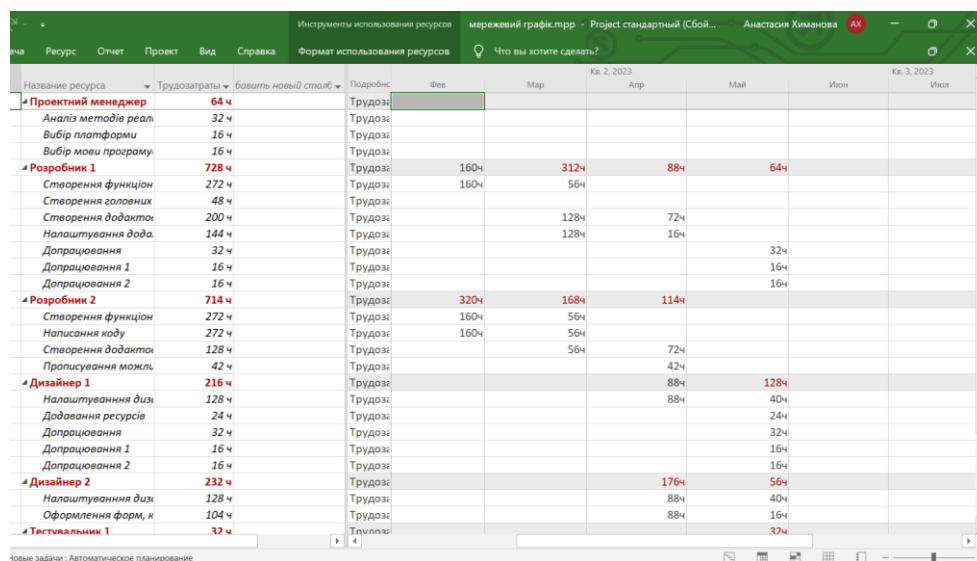


Рисунок 2.2 - Використання ресурсів

Джерело: запропоновано автором

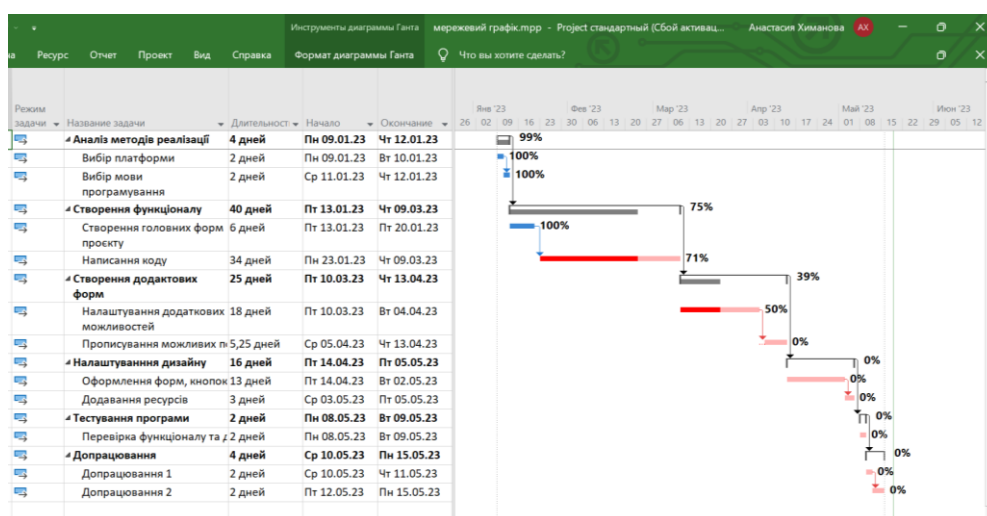


Рисунок 2.3 - Діаграма Ганта з відслідковуванням

Джерело: запропоновано автором

Можемо зробити висновок, що діаграма Ганта є універсальним інструментом, який допомагає в будь-якому плануванні, має безліч додаткових можливостей, допомагає отримати загальну картину того, як завдання еволюціонують з часом.

Дуже корисно використовувати планування при виконанні будь яких завдань, бо відразу видно які дії за чим виконуються, хто за що відповідає, та взагалі якою буде протяжність проекту.

2.3 Проектування бази даних

База даних в даному проекті є важливою ігровою механікою, яка додає реалістичності та деталізації до симуляції діяльності фірми. Ось кілька причин, чому вона є необхідною:

- Зберігання історії проектів: база даних дозволяє зберігати інформацію про різні проекти, їх стан, хід виконання, рівень успішності тощо, це дозволяє користувачам відстежувати прогрес кожного проекту і аналізувати результати.
- Управління ресурсами: база даних зберігає інформацію про співробітників, їх навички та зарплати, це дозволяє користувачам вибирати програмістів, дизайнерів, тестувальників і т.д. відповідно до їхніх вимог та фінансових можливостей.
- Відстеження фінансових показників: база даних зберігає інформацію про витрати на кожного співробітника, включаючи їхню зарплатню, користувачі можуть відстежувати загальні витрати та аналізувати ефективність вкладення коштів у проекти.
- Запис результатів гри: база даних зберігає дані, отримані в ході міні-ігор, таких як рахунок гравця та час гри, це дозволяє користувачам порівнювати результати, визначати кращих гравців і стимулювати змагальний дух.
- Аналітика та статистика: зібрані дані в базі даних можуть бути використані для аналізу та створення статистичних звітів, користувачі можуть оцінювати ефективність проектів, розраховувати середні показники та виявляти тенденції, що допомагають в управлінні та прийнятті рішень.
- Зручний доступ до даних: база даних забезпечує централізований доступ до всієї інформації про проекти, співробітників та результати гри, це робить

систему більш зручною для користувачів і допомагає уникнути втрати або дублювання даних.

Загалом, база даних в цьому проекті допомагає створити комплексну інформаційну систему, яка зберігає, аналізує та використовує дані для покращення управління та взаємодії з користувачами. Вона робить симуляцію фірми більш реалістичною, інтерактивною та корисною для освітніх цілей.

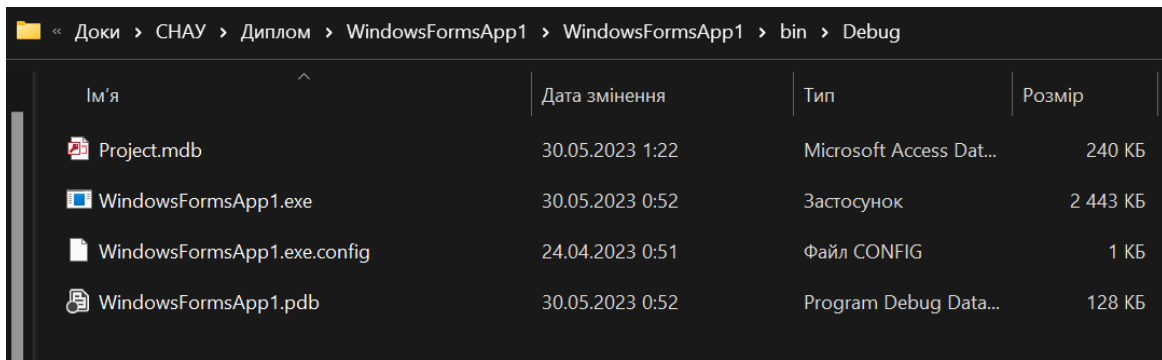
Процес створення бази даних для навчально-інформаційної системи-симулятора діяльності фірми, створювали ми її за допомогою програми Microsoft Access. Для початку ми створили саму базу під назвою Project та таблицю в ній під назвою project, додавши до неї потрібні нам поля в режимі конструктор, та налаштувавши тип цих полів, наведено на рисунку 2.4.

Ім'я поля	Тип даних
w_id	Автонумерація
w_name	Короткий текст
w_programmer	Короткий текст
w_programmer_salary	Грошова одиниця
w_designer	Короткий текст
w_designer_salary	Грошова одиниця
w_tester	Короткий текст
w_tester_salary	Грошова одиниця
w_days	Число
w_budget	Грошова одиниця
w_price	Грошова одиниця
w_money_left	Грошова одиниця

Рисунок 2.4 - Конструктор таблиці бази даних

Джерело: запропоновано автором

При збереженні було замінено стандартне розширення бази даних .accdb на .mdb і головною умовою для правильного підключення є правильне розміщення файлу в проекті, це наведено на рисунку 2.5.



Ім'я	Дата змінення	Тип	Розмір
Project.mdb	30.05.2023 1:22	Microsoft Access Dat...	240 КБ
WindowsFormsApp1.exe	30.05.2023 0:52	Застосунок	2 443 КБ
WindowsFormsApp1.exe.config	24.04.2023 0:51	Файл CONFIG	1 КБ
WindowsFormsApp1.pdb	30.05.2023 0:52	Program Debug Data...	128 КБ

Рисунок 2.5 - Розміщення бази даних

Джерело: запропоновано автором

Для зв'язку проекту з базою даних були використані SQL-запити, рядок підключення, взаємодію, розглянуто в наступному розділі.

РОЗДІЛ 3

РОЗРОБКА ІГРОВОГО СИМУЛЯТОРА

3.1 Архітектура додатку

Архітектура додатку "Навчально інформаційна система-симулятор діяльності фірми" базується на популярному шаблоні проектування Model-View-Controller (MVC). Цей шаблон відокремлює логіку додатку від його представлення та спрощує його розширення та підтримку, представлений на рисунку 3.1.



Рисунок 3.1 – Шаблон MVC

Джерело: запропоновано автором

У контексті цього додатку, модель (Model) включає базу даних Microsoft Access і відповідає за зберігання та оновлення всіх даних, пов'язаних з проектами, програмістами, дизайнерами, тестувальниками тощо. Модель виконує запити до бази даних за допомогою SQL-запитів, отримує, оновлює та видаляє записи, а також забезпечує цілісність та безпеку даних.

Представлення (View) відповідає за графічний інтерфейс користувача. За допомогою Windows Forms, розробники створюють вікна, кнопки, текстові поля та

інші елементи, які користувач використовує для взаємодії з додатком. Представлення відображає дані з моделі користувачеві та передає його дії до контролера.

Контролер (Controller) виконує роль посередника між моделлю та представленням. Він приймає події, спричинені діями користувача, такі як натискання кнопок чи введення тексту, та ініціює відповідні дії. Контролер отримує дані з моделі, оновлює їх або передає команди моделі для виконання необхідних операцій. Він також оновлює представлення, щоб відобразити зміни, які сталися в результаті взаємодії з користувачем.

База даних Microsoft Access є важливою складовою цієї архітектури. Вона забезпечує зручну збереження та отримання даних про проекти, програмістів, дизайнерів, тестувальників та інші. База даних містить таблиці, які відображають основні сутності додатку та їх взаємозв'язки. Через використання SQL-запитів, модель може взаємодіяти з базою даних, виконуючи пошук, вставку, оновлення та видалення даних.

Крім того, після кожного етапу, такого як програмування, дизайн або тестування, додаток виводить повідомлення з кількістю витрачених днів та коштів. Це дає користувачам зрозуміти, скільки ресурсів було витрачено на кожний етап проекту. Після завершення всього проекту також виводиться повідомлення з інформацією про загальну кількість днів та коштів, витрачених на проект у цілому.

В кінці додатку знаходиться головне меню, яке дозволяє користувачам почати виконання нового проекту або переглянути дані про виконані проекти, збережені в базі даних. Це забезпечує зручний спосіб взаємодії з додатком та доступ до інформації про минулі проекти.

Таким чином, архітектура додатку "Навчально-інформаційна система-симулятор діяльності фірми" забезпечує ефективну організацію та взаємодію компонентів додатку, забезпечуючи користувачам зручну роботу з системою та збереження важливої інформації в базі даних Microsoft Access.

3.2 Розробка симулятору виконання проектів

Початковим етапом розробки проекту було визначення вимог до системи. Згідно з метою кваліфікаційної роботи, було з'ясовано, що система повинна надавати можливість користувачам створювати та керувати проектами, вибирати команду програмістів, дизайнерів і тестувальників, а також здійснювати моніторинг витрат часу та коштів.

Для реалізації цих вимог було обрано середовище розробки Visual Studio та мову програмування C#. Застосування Windows Forms дало змогу створити графічний інтерфейс користувача, який був зручним у використанні та надавав необхідні функції для взаємодії з додатком.

Всі форми проекту є значущими та взаємодіють одна з одною, розглянемо деякі з них. Першою ми створили форму `main_form`, вона представлена на рисунку 3.2, додавши до неї один елемент `label` та чотири `button`.

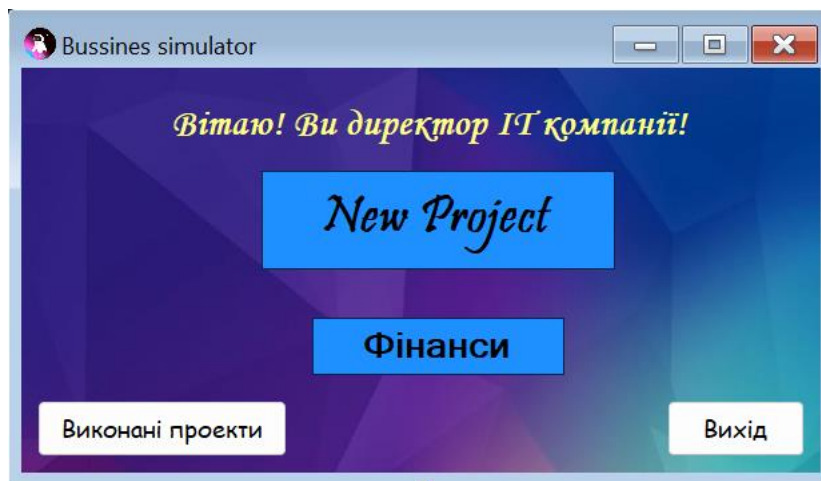


Рисунок 3.2 - Вигляд форми `main_form`

Джерело: запропоновано автором

Код елементів форми: кнопка «вихід» організовує просто закриття форми та вихід з програми:

```
Form mainForm = this.FindForm();  
mainForm.Close();
```

```
Application.Exit();
```

Кнопка «Фінанси» направляє нас на сторінку html де ми можемо ознайомитися з фінансовим менеджментом, це описано в наступному розділі.

Кнопка «Виконані проекти» відбувається перехід на форму переглядів виконаних проектів (цю форму ми розглянемо пізніше):

```
project projectForm = new project();
projectForm.Show();
this.Hide();
```

Далі кнопка «New Project» на ній відбувається перехід на наступну форму з введенням назви проекту, це наведено на рисунку 3.3.

```
project_name nameForm = new project_name();
nameForm.Show();
this.Hide();
```

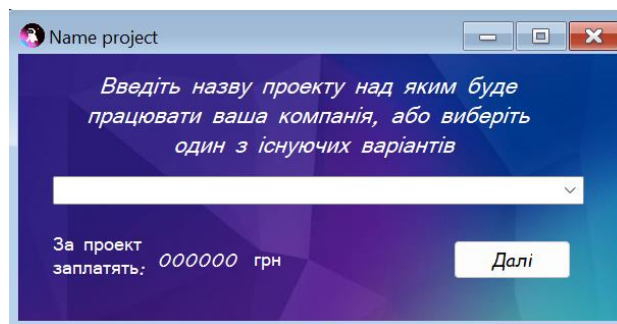


Рисунок 3.3 - Вигляд форми project_name

Джерело: запропоновано автором

На даній формі можна вписати назву проекту самостійно, або вибрати одну з запропонованих, після вибору наведено на рисунку 3.4.

При натисканні кнопки не просто відбувається перехід на наступну форму, а назва проекту, та ціна за проект записуються в базу даних, також id рядка запам'ятовується, та передається наступній формі, щоб дані по проекту записували в той самий рядок, а не створювався наступний:

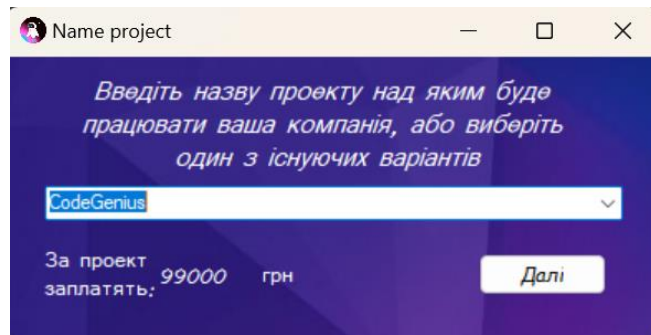


Рисунок 3.4 – Вибір проекту

Джерело: запропоновано автором

```
// Створення нового запису в базі даних та перехід до форми programmer
connection.Open();
OleDbCommand cmd = new OleDbCommand("SELECT COUNT(*) FROM
project", connection);
int count = (int)cmd.ExecuteScalar();
int w_id = count + 1;
string w_name = comboBox1.Text;
int colonIndex = w_name.IndexOf(':');
if (colonIndex > 0)
{
    w_name = w_name.Substring(0, colonIndex).Trim();
}
cmd.CommandText = "INSERT INTO project (w_id, w_name) VALUES (" +
w_id + ", '" + w_name + "')";
cmd.ExecuteNonQuery();
connection.Close();
// Перехід на наступну форму
Choice choiseForm = new Choice(w_id);
choiseForm.Show();
this.Hide();
```

Далі імпровізована ігрова форма на якій потрібно вгадати який етап йде наступним, представлено на рисунку 3.5.

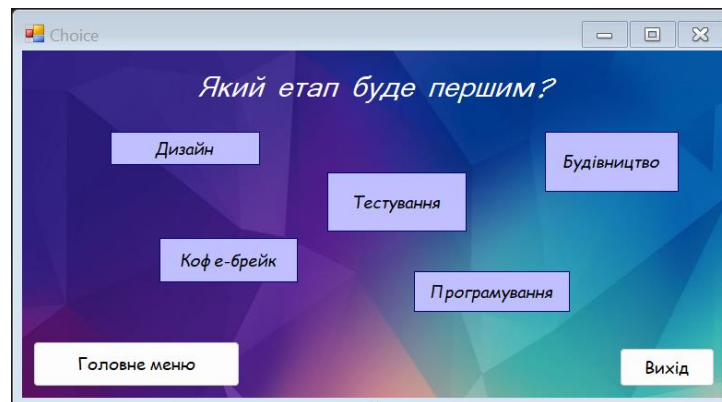


Рисунок 3.5 - Форма вибору наступного етапу

Джерело: запропоновано автором

Правильним варіантом є кнопка «програмування», при натисканні на неї відбувається перехід на форму вибору програміста, а при виборі інших варіантів, ми бачимо повідомлення по типу:

```
MessageBox.Show("Цей етап буде пізніше!", "Помилка!");
```

Наступна форма вибору програміста, представлена на рисунку 3.6 по її зразку зроблено і форми вибору дизайнера та тестувальника.

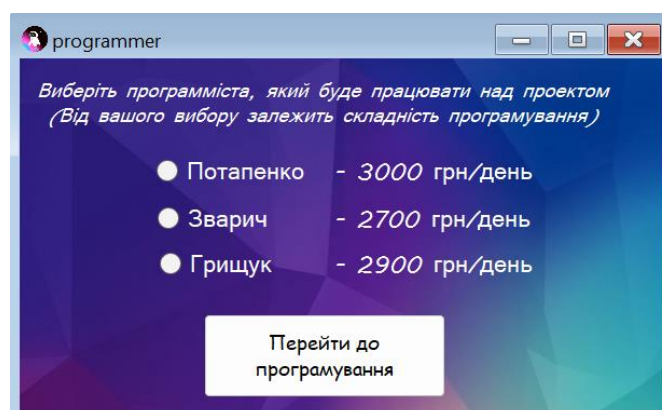


Рисунок 3.6 - Форма вибору програміста

Джерело: запропоновано автором

На формі розміщений елемент label з текстом, а також три елементи radioButton для вибору, та кнопка button за допомогою якої наші дані записуються в базу даних та відбувається перехід до наступної форми з запам'ятовуванням вибору:

// Додавання даних про програміста та його зарплатні до запису в базі даних

```
string programmer = "";
int salary = 0;
if (radioButton1.Checked)
{
    programmer = radioButton1.Text;
    salary = 3000;
}
else if (radioButton2.Checked)
{
    programmer = radioButton2.Text;
    salary = 2700;
}
else if (radioButton3.Checked)
{
    programmer = radioButton3.Text;
    salary = 2900;
}
```

// Оновлюємо запис в базі даних

```
string connectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=Project.mdb;";
using (OleDbConnection connection = new
OleDbConnection(connectionString))
{
    connection.Open();
```

```
OleDbCommand command = new OleDbCommand("UPDATE project SET
w_programmer = @programmer, w_programmer_salary = @salary WHERE w_id =
(SELECT MAX(w_id) FROM project)", connection);

command.Parameters.AddWithValue("@programmer", programmer);
command.Parameters.AddWithValue("@salary", salary);
command.ExecuteNonQuery();
}
```

MessageBox.Show("Гра П'ятнашки, вам потрібно розставити всі цифри в вірному порядку, в цей час будуть рахуватися дні виконання (10 секунд вважається як 1 день) Бажаю успіху!", "Інструкція");

Ми розробили ігрові форми для кожного етапу, як імітація цих дій, для програмування це гра «п'ятнашки» в якій є різні рівні складності в залежності від вибору програміста, сам конструктор форми має тільки два елементи представлені на рисунку 3.10, один текстовий label та одна кнопка button при натисканні на яку відбувається перемішування клітинок, також на формі представлений таймер, в якому стоїть параметр оновлення кожні 10 секунд, тобто кожні 10 секунд елемент label змінюється на +1, після завершення завдання (в нашому випадку правильного розставлення цифр) зарплата вибраного робітника перемножується на кількість витрачених днів, та записується в базу даних, а також виводиться в повідомленні.

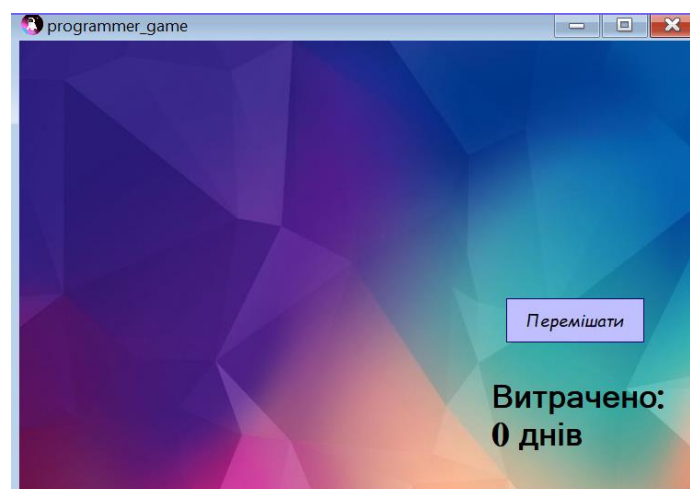


Рисунок 3.7 - Ігрова форма програмування

Джерело: запропоновано автором

Код форми, спочатку визначається кількість квадратів та їх розмір в залежності від параметра salary тобто зарплати вибраного програміста:

```
if (salary == 3000)
{
    BOARD_SIZE = 3;
    SQUARE_SIZE = 80;
}
else if (salary == 2700)
{
    BOARD_SIZE = 5;
    SQUARE_SIZE = 57;
}
else if (salary == 2900)
{
    BOARD_SIZE = 4;
    SQUARE_SIZE = 70;
}
```

Вигляд форми після запуску гри з параметром (salary == 3000) наведено на рисунку 3.8.

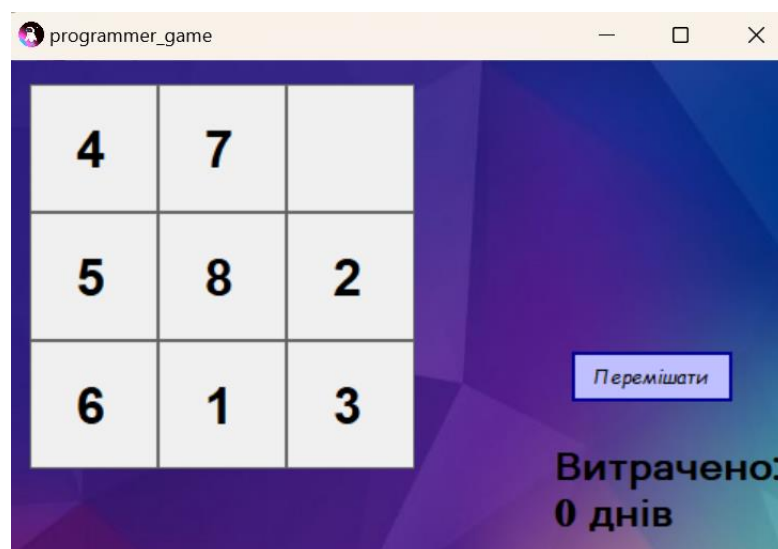


Рисунок 3.8 – Вигляд запущеної форми імітації програмування

Джерело: запропоновано автором

Логіку самої гри представлено в додатку Б.

По типу даної форми також зроблена форма тестування, на ній представлена гра «хрестики нолики», вигляд форми до запуску гри наведений на рисунку 3.9.

Вигляд форми після запуску гри з параметром (`salary == 1800`) наведено на рисунку 3.10.

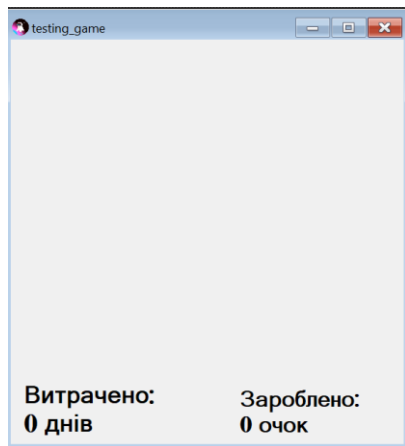


Рисунок 3.9 - Ігрова форма тестування

Джерело: запропоновано автором

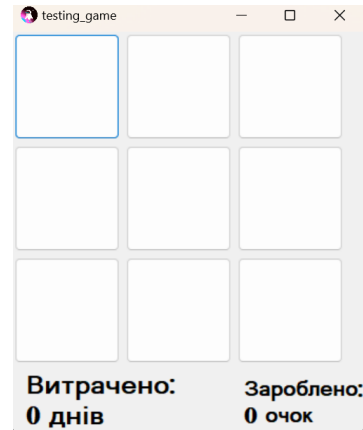


Рисунок 3.10 – Вигляд запущеної форми імітації тестування

Джерело: запропоновано автором

В імітації тестування інша логіка самої гри, потрібно виграти не лише один раз, а 10 разів, після кожного виграшу дані записуються в елемент label, а дні записуються так само тільки не кожні 10 секунд, а кожні 5 секунд.

Наступна форма дизайну, представлена на рисунку 3.11, вона має трішки інший підхід, на ній організована гра «Клон OSU».

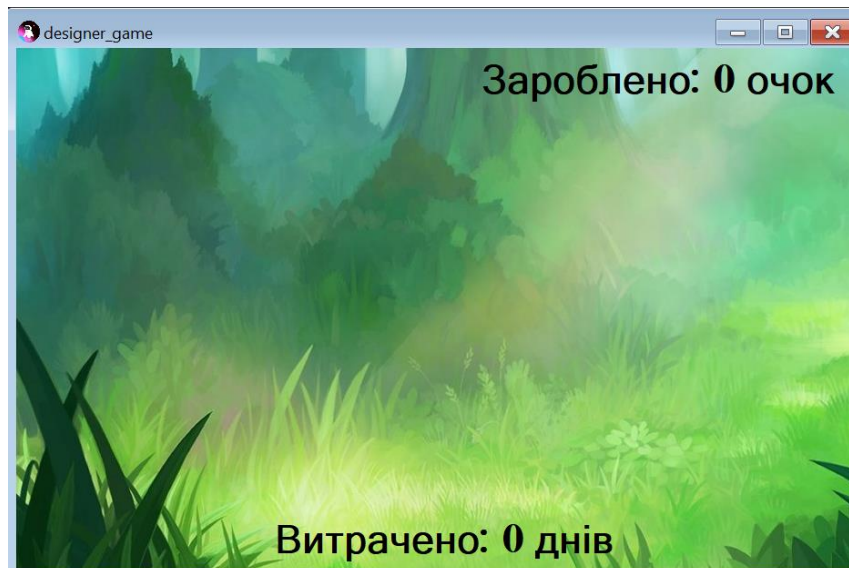


Рисунок 3.11 - Ігрова форма дизайну

Джерело: запропоновано автором

Для створення даної гри був доданий файл ресурсів, та до нього додані елементи такі, як: картинка заднього плану, сам круг який рухатиметься по формі, та контур яким ловиться круг, це представлено на рисунку 3.12.

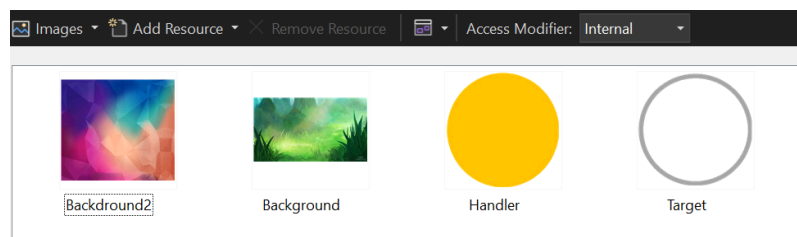


Рисунок 3.12 - Ігрова форма дизайну

Джерело: запропоновано автором

На даній формі встановлений таймер, 5 секунд рахується як один день, це значення відображається на формі, головним завданням гри є попадання контуром рівно в круг, при такому попаданні зараховується одне очко, для виграшу потрібно набрати 20 очок, це теж відображається на формі.

Після завершення всіх етапів імітації розробки проекту виводиться повідомлення з інформацією про ресурси витрачені на нього, це представлено на рисунку 3.13.

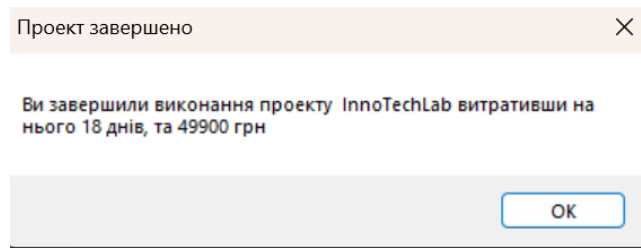


Рисунок 3.13 – Повідомлення по завершенню проекту

Джерело: запропоновано автором

Далі йде форма, на якій виводяться дані з бази даних такі, як: ціна проекту, витрати на розробку та залишок коштів, це представлено на рисунку 3.14.

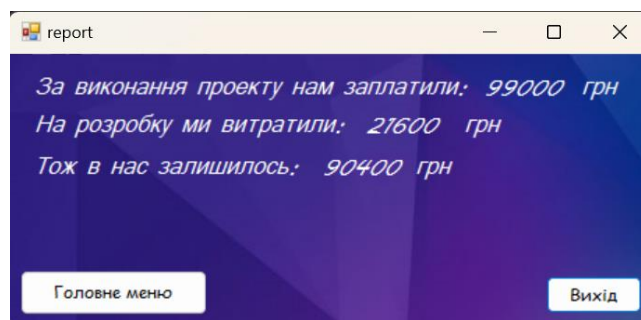


Рисунок 3.13 – Вигляд форми report

Джерело: запропоновано автором

Після завершення проекту було реалізовано можливість перегляду виконаних проектів через головне меню додатку. Користувачі мали змогу переглянути дані про кожен проект, включаючи кількість днів та коштів, витрачених на кожен етап, а також загальну інформацію про витрати. На формі є три елементи button та один елементом listView в якому виводиться інформація про виконані проекти, це наведено на рисунку 3.17.

Код для виведення інформації з бази даних на форму:

// Підключення до бази даних Access

```
string connectionString = @"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=Project.mdb";

using (OleDbConnection connection = new
OleDbConnection(connectionString))
```

```

{
    // Запит для отримання даних з таблиці
    string query = "SELECT w_name, w_programmer, w_designer, w_tester,
w_days, w_budget FROM project";

    // Створення команди
    OleDbCommand command = new OleDbCommand(query, connection);

    // Відкриття з'єднання з базою даних
    connection.Open();

    // Виконання команди та отримання результату
    OleDbDataReader reader = command.ExecuteReader();

    // Очищення ListView
    listView1.Items.Clear();
    listView1.Columns.Clear();

    // Налаштування властивості View на View.Details
    listView1.View = View.Details;

    // Додавання стовпців до ListView
    listView1.Columns.Add("w_name", 80);
    listView1.Columns.Add("w_programmer", 80);
    listView1.Columns.Add("w_designer", 80);
    listView1.Columns.Add("w_tester", 80);
    listView1.Columns.Add("w_days", 50);
    listView1.Columns.Add("w_budget", 70);

    // Додавання рядків до ListView
    while (reader.Read())
    {
        ListViewItem item = new ListViewItem(reader["w_name"].ToString());
        item.SubItems.Add(reader["w_programmer"].ToString());
        item.SubItems.Add(reader["w_designer"].ToString());
        item.SubItems.Add(reader["w_tester"].ToString());
        item.SubItems.Add(reader["w_days"].ToString());
    }
}

```

```

        item.SubItems.Add(reader["w_budget"].ToString() + " грн");
        listView1.Items.Add(item);
    }
    // Закриття з'єднання з базою даних
    connection.Close();
}

```

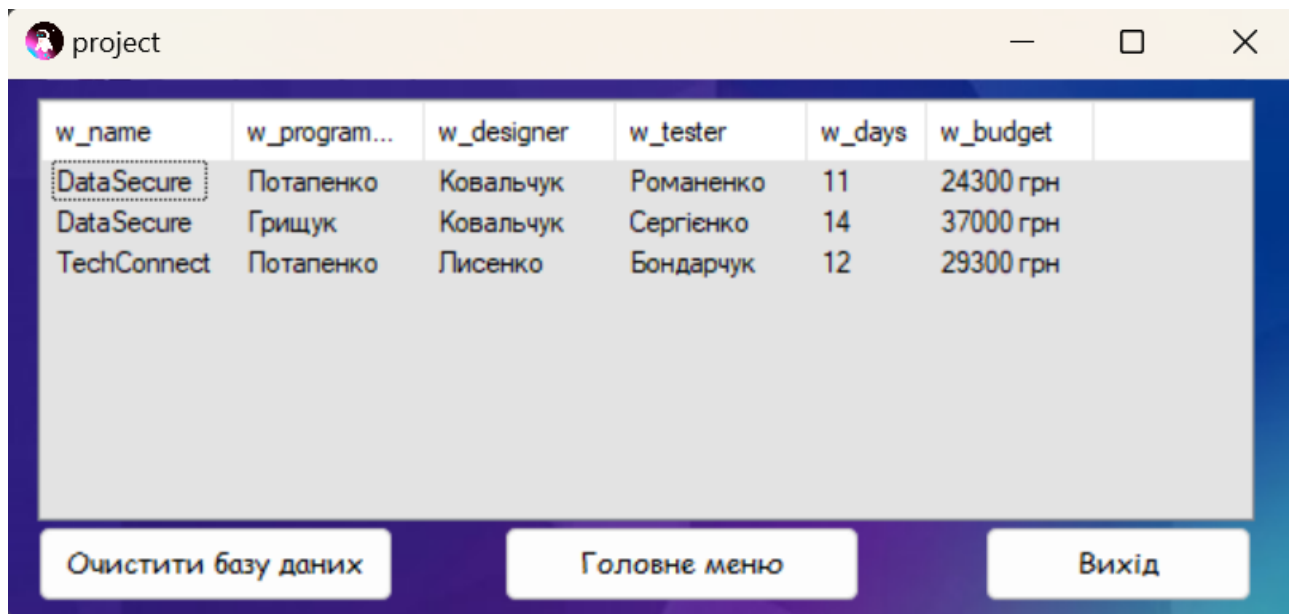


Рисунок 3.17 – Вигляд форми виконаних проектів

Джерело: запропоновано автором

Під час розробки було враховано основні вимоги до функціональності системи. Було створено головне меню з кнопками для перегляду виконаних проектів та створення нових проектів. Крім того, були реалізовані різні етапи роботи, включаючи програмування, дизайн та тестування, кожен з яких мав свої відповідні міні-ігри для вимірювання часу та результативності виконавців.

Результатом роботи є функціональний додаток, який дозволяє користувачам відтворювати та керувати робочими процесами фірми. Додаток надає зручний інтерфейс та можливість вимірювання часу та результативності виконавців на міні-іграх. Крім того, база даних забезпечує зберігання та доступ до даних про виконані проекти.

3.2 Розробка симулятора фінансових операцій

Основна мета розробки симулятора фінансових операцій полягає у наданні користувачеві можливості вивчити та вдосконалити свої навички у фінансових операціях без реальних фінансових ризиків. Він може використовуватися студентами, трейдерами, інвесторами або будь-якою особою, яка цікавиться фінансовими ринками.

За допомогою симулятора фінансових операцій користувач може:

- Взяти позику в банку під певний відсоток.
- Купувати та продавати акції різних компаній.
- Спостерігати за змінами цін на акції та вивчати ринкові тенденції.
- Аналізувати та оцінювати ефективність своїх інвестиційних стратегій.
- Повертати запозичені кошти та здійснювати інші фінансові операції.

Такий симулятор дозволяє користувачам отримати практичний досвід у фінансових операціях, розвивати аналітичні та прийняття рішень навички, а також тестувати свої ідеї та стратегії, не ризикуючи реальними грошима.

Натиснувши кнопку «фінанси» на головній формі симулятору управління проектами, відкривається головна сторінка симулятора фінансових операцій, це наведено на рисунку 3.18.

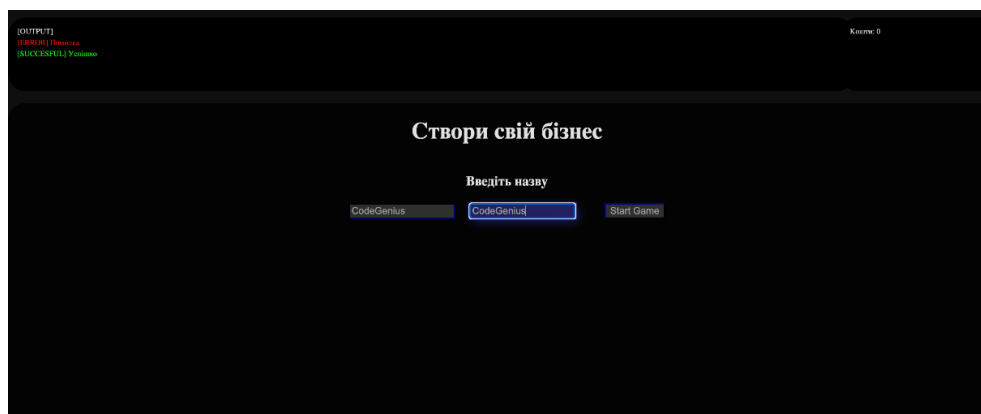


Рисунок 3.18 – Головна сторінка

Джерело: запропоновано автором

В поле назви вписується назва проекту яким ми перед цим виконували в додатку для Windows.

Далі наступна сторінка з можливістю взяти позику в банку під вибраний відсоток, можна обрати один з запропонованих тарифів, представлено на рисунку 3.19.

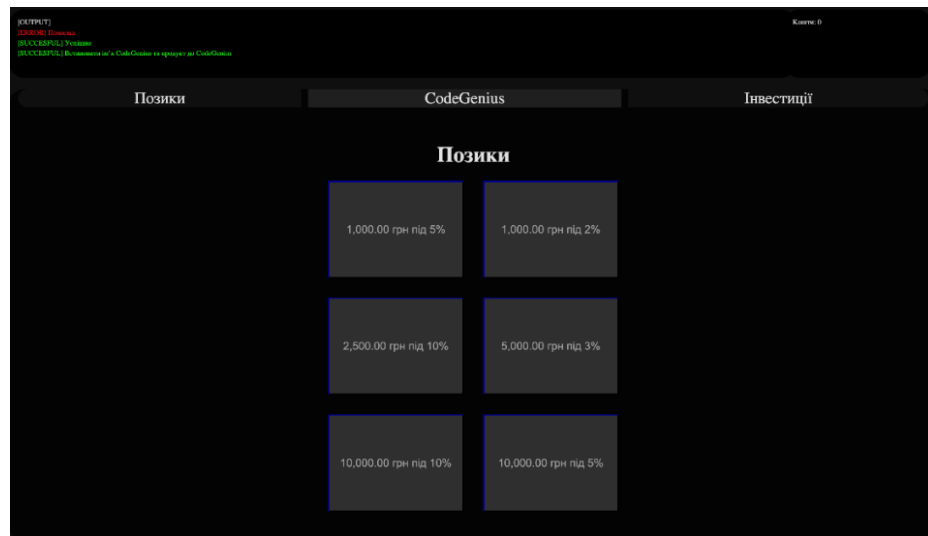


Рисунок 3.18 – Вибір тарифу позики

Джерело: запропоновано автором

Нижче наведений код вкладки отримання позики:

```
<h1 class="title">Позики</h1>
```

```
<div id="loanbuttons">
```

```
  <button onclick="setLoan(100000,.05,true,60000)">1,000.00 грн під  
5%</button>
```

```
  <button onclick="setLoan(100000,.02,false,60000)">1,000.00 грн під  
2%</button>
```

```
  <button onclick="setLoan(250000,.1,true,60000)">2,500.00 грн під 10%  
</button>
```

```
  <button onclick="setLoan(500000,.03,false,60000)">5,000.00 грн під 3%  
</button>
```

```
  <button onclick="setLoan(1000000,.1,true,60000)">10,000.00 грн під 10%
```

```
</button>
```

```
<button onclick="setLoan(1000000,.05,false,60000)">10,000.00 грн під 5%
```

```
</button>
```

```
</div>
```

```
<div id="afterloan" style="display: none;">
```

```
<input type="text" id="afterloan-payamm"></input>
```

```
<button onclick='payLoan(document.getElementById("afterloan-payamm").value)'>Розплатитися</button>
```

```
</div>
```

Далі присутній функціонал для роботи з фондовим ринком для купівлі та продажу акцій компаній, наприклад, вибираємо одну з компаній, пишемо кількість акцій яку хочемо купити та натискаємо кнопку покупка, після цього слідкуємо ціна на акції падає чи піднімається, коли ми вважаємо що акція достатньо зросла, то можемо продати її, аби отримати кошти, це представлено на рисунку 3.19.

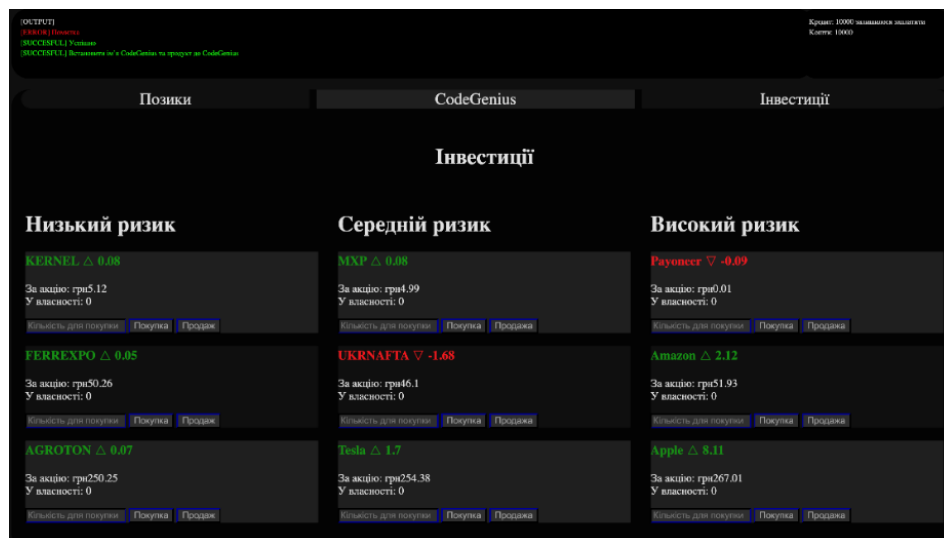


Рисунок 3.19 – Фондовий ринок

Джерело: запропоновано автором

Нижче представлено код відображення декількох акцій компаній, а повний код, наведено в додатку Б.

```
<div style="display: none" class="menu">
```

```
<h1 class="title">Інвестиції</h1>
```

```

<div id="stockscontainer">
  <div id="lowrisk">
    <h1>Низький ризик</h1>
    <div class="stock">
      <h3>KERNEL – 0</h3>
      <p>За одну акцію: 10 грн <br> У власності: 0</p><input type="text"
id="stock1" placeholder="Кількість для покупки">
      <button
onclick="BuyStock(0,1,document.getElementById('stock1').value)">Покупка</button>
    >
      <button onclick="BuyStock(0,1,-
document.getElementById('stock1').value)">Продаж</button>
    </div>
    <div class="stock">
      <h3>FERREXPO – 0</h3>
      <p>За одну акцію: 100 грн <br> У власності: 0</p><input type="text"
id="stock2" placeholder="Кількість для покупки">
      <button
onclick="BuyStock(0,2,document.getElementById('stock2').value)">Покупка</button>
    >
      <button onclick="BuyStock(0,2,-
document.getElementById('stock2').value)">Продаж</button>
    </div>
    <div class="stock">

```

Після того як ми заробили достатню суму коштів, переходимо на вкладку повернення, вводимо суму яку хочемо повернути в банк, та натискаємо кнопку розплатитися, це представлено на рисунку 3.20.

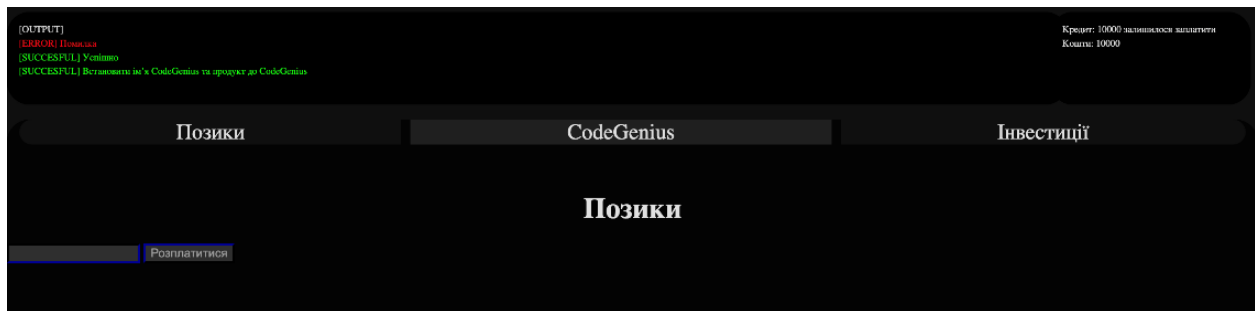


Рисунок 3.20 – Повернення позики

Джерело: запропоновано автором

Нижче представлений код повернення позики:

```
const payLoan = (quantity) => {
const asInt = parseInt(quantity);
if (isNaN(asInt)) {
  logError(`${quantity} не є числом`);
  return;
}
const payment = asInt * 100;
if (funds >= payment) {
  addFunds(-payment);
  loan.amount -= payment;
  loanHTML.innerText = `Кредит: ${loan.amount} залишилось заплатити`;
  logSuccess(`Сплачено ${quantity}`);
} else {
  logError(`у вас недостатньо ${quantity} тисяч`);
}
};
```

В даному розділі було розглянуто розробку симулятора управління проектами, який допомагає розібратися в тому які етапи йдуть за одним, та розробку симулятора фінансових операцій, який надає можливість трішки ознайомитися з фондовим ринком.

ВИСНОВОК

У ході розробки дипломної роботи була створена навчально-інформаційна система-симулятор діяльності фірми, яка надає можливість користувачам відтворити та керувати процесами, що відбуваються в організації. Основними функціональними елементами системи є головне меню, HTML-сторінка з симулятором, можливість отримання позики в банку під відсоток та вкладання коштів в акції.

Архітектура додатку, побудована за допомогою шаблону проектування Model-View-Controller (MVC), забезпечує відокремлення логіки додатку, графічного інтерфейсу та бази даних, що сприяє зручній розширюваності та підтримці системи. Використання середовища Visual Studio та мови програмування C# разом із Windows Forms надають зручний інтерфейс користувача та ефективну роботу з додатком.

Завдяки розробленому додатку, користувачі можуть отримати практичні навички в управлінні проектами та розуміння робочих процесів в сфері комп'ютерних наук. Дана навчально-інформаційна система-симулятор може бути використана в навчальних закладах для практичного застосування теоретичних знань у сфері управління проектами.

Розроблений веб-додаток симулятор фінансових операцій, що забезпечує імітацію купівлі-продажу акцій підприємств. Веб-додаток може використовуватися трейдерами та інвесторами або будь-якою особою, яка цікавиться фінансовими ринками. Такий симулятор дозволяє користувачам отримати практичний досвід у фінансових операціях, розвивати аналітичні та прийняття рішень навички, а також тестувати свої ідеї та стратегії, не ризикуючи реальними грошима.

Отже, розроблена навчально-інформаційна система-симулятор діяльності фірми дозволяє користувачам практично ознайомитись з процесами, що відбуваються в компанії. Ця система надає можливість ефективно управляти фінансами, вибирати команду розробників та отримувати звіти про результати кожного проекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гайдаржи В.І., Ізварін І.В. Бази даних в інформаційних системах. К.; Університет «Україна», 2018. 418 с. (дата звернення 08.06.2023)
2. Єгорченкова Н. Ю., Єгорченков О. В., Катаєва Є. Ю. Азбука управління проектами. Плануванн: навч. посіб. Київ: КНУ ім.Т.Шевченка, 2017. – 91с.
3. Концедайло В. В., “Використання серйозних ігор та симуляцій з розробки програмного забезпечення для розвитку нетехнічних компетентностей майбутніх інженерів-програмістів”, Наукові записки, Серія: проблеми методики фізико-математичної і технологічної освіти, №1, 2017. – 47 - 53 с.
4. Офіційний веб-сайт Фонду вільного програмного забезпечення (Free Software Foundation - FSF) URL: <https://www.gnu.org/licenses/lgpl-3.0.html> (дата звернення: 23.05.2023).
5. Про затвердження інструкції з ведення погосподарського обліку в сільських, селищних та міських радах. URL: <https://zakon.rada.gov.ua/laws/show/z0689-16#Text> (дата звернення 08.06.2023)
6. Стратегія і основні кроки при розробці web-сайту URL: <http://ruszura.in.ua/neobhidno-znaty/stratehiya-s-osnovns-kroky> pryrozrobsiweb-sajta.HTML (дата звернення:08.06.2023)
7. Хіманова А. О. Використання симуляторів у освітній діяльності університету / Технології ХХІ сторіччя: Збірник тез за матеріалами 28-ої міжнародної науково – практичної конференції (23-25 листопада 2022 р.). Ч.2. – Суми: СНАУ, 2022 - с.184.
8. Хіманова А. О. Проблеми управління іт проектами сучасних навчальних систем / Матеріали науково-практичної конференції викладачів, аспірантів та студентів Сумського НАУ (25-28 квітня 2023 р.). – Суми, 2023. – с.355.
9. Цеслів О. В., Коломієць А.С. Технологія проєктування та адміністрування баз даних і сховищ даних : навч. посібник; Нац. техн. ун-т України «Київ. політехн. ін-т ім. Ігоря Сікорського». Київ : КПІ ім. І. Сікорського : Політехніка, 2017. 281 с. (дата звернення 08.06.2023)

10. Шаховська Н. Б., Камінський Р.М., Вовк О.Б. Системи штучного інтелекту : навч. посібник Львів : Львівська політехніка, 2018. 391 с. (дата звернення 08.06.2023)
11. Що таке веб додаток? Різниця між сайтом, веб-додатком, spa і pwa URL: <https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/> (дата звернення: 08.06.2023)
12. HTTP (Hypertext Transfer Protocol) URL: <https://www.techtarget.com/whatis/definition/HTTP-Hypertext-Transfer-Protocol> (дата звернення:08.06.2023)
13. Institute P. M. Guide to the Project Management Body of Knowledge (PMBOK® Guide)-Sixth Edition (GERMAN). Project Management Institute, 2017. – 43с.
14. JavaScript URL: <https://uk.wikipedia.org/wiki/JavaScript> (дата звернення:08.06.2023)
15. Kerzner H. Project Management: A Systems Approach to Planning, Scheduling, and Controlling. Wiley, 2017. – 38 с.
16. Michael, D., Chen, S., Alavi, M. Business simulation games: Current usage levels and recommendations for best practices. Journal of Education for Business, 91(6), 2016. – 317 - 325 с.
17. Miles R. C# Programming Yellow Book: Learn to Program in C# from First Principles. Independently Published, 2018. – 10 с.
18. Visual Studio Code. URL: https://ru.wikipedia.org/wiki/Visual_Studio_Code (дата звернення:08.06.2023)

ДОДАТКИ

ТЕХНІЧНЕ ЗАВДАННЯ

на «створення навчально інформаційної системи-симулятора
діяльності фірми»

ПОГОДЖЕНО:

доцент кафедри кібернетики та інформатики

Студентка групи _____

(ПБ)

Суми 2023

1. Призначення й мета створення навчально інформаційної системи-симулятора діяльності фірми

1.1 Призначення системи

Інформаційна система має навчати та надавати розуміння основних аспектів бізнес-процесів, прийняття фінансових рішень, та їх впливу на фінансові показники та кінцеві результати діяльності компанії.

1.2 Мета створення інформаційної системи

Надання користувачам можливості отримати практичні навички управління фірмою та розуміння впливу фінансових рішень на кінцеві результати.

1.3 Цільова аудиторія

Цільовою аудиторією навчально-інформаційної системи-симулятора діяльності фірми можуть бути студенти, підприємці, менеджери та будь-які особи, зацікавлені у вивченні та розвитку навичок управління фірмою.

2. Вимоги до створення навчально інформаційної системи симулятора діяльності фірми

2.1 Вимоги до інформаційної системи в цілому

2.1.1 Вимоги до структури й функціонування інформаційної системи

Інформаційна система повинна бути доступною та зрозумілою для користувачів.

2.1.2 Вимоги до персоналу

Від персоналу вимагаються загальні навички роботи з персональним комп'ютером, а також знання таких мов програмування, як: C#, HTML, CSS, JavaScript, хоча б на початковому рівні

2.1.3 Вимоги до збереження інформації

Уся інформація буде зберігатися в базі даних реалізований в програмному додатку Microsoft Access.

2.2 Структура інформаційної системи

2.2.1 Загальна інформація про структуру системи

Структура навчально-інформаційної системи-симулятора діяльності фірми повинна бути організована наступним чином:

1. Головне меню: Вхідна точка системи, де користувачі мають доступ до основних функцій та можливостей системи.
2. HTML-сторінка: На цій сторінці користувачі можуть взяти позику в банку під відсоток та вкладати кошти в акції.
3. Симулятор діяльності фірми: В цьому модулі користувачі взаємодіють з віртуальною фірмою, виконують дії, такі як створення нового проекту, вибір персоналу (розробників, дизайнерів, тестувальників) і проведення міні-ігор.
4. Міні-ігри: дозволяють користувачам заробляти гроші та розвивати навички.
5. База даних: Записує та зберігає дані про дії користувачів, фінансові показники, витрати на персонал та іншу важливу інформацію.

2.2.2 Навігація

В нашій системі всі блоки йдуть один за одним, тому майже нема потреби в навігації, та майже з кожного блоку можна повернутися в головне меню.

2.2.3 Дизайн та структура системи

Дизайн та структура навчально-інформаційної системи-симулятора діяльності фірми мають на меті забезпечити зручну та інтуїтивно зрозумілу взаємодію для користувачів. Основні принципи дизайну та структури повинні включати:

1. Головне меню: Вхідна точка системи, де користувачі зустрічаються з основними опціями та функціями системи. Дизайн головного меню повинен бути простим, зрозумілим і естетично привабливим, щоб користувачі могли легко навігуватись і знаходити потрібні опції.
2. Розміщення елементів: Елементи і функції системи, такі як кнопки, посилання та інтерактивні елементи, повинні бути розміщені в логічному та легкодоступному порядку. Наприклад, опції взяття позики в банку або вкладання коштів можуть бути представлені в окремій секції або пункті меню.

3. HTML-сторінка: Дизайн сторінки, де доступні опції позики та інвестування, повинен бути зручним для користувачів. Інтерфейс може містити форми для введення даних, кнопки для підтвердження дій та відображення розрахунків.

4. Симулятор діяльності фірми: Дизайн симулятора має передавати відчуття управління фірмою та бізнес-середовища. Він може включати графічні елементи, ілюстрації, діаграми та інші візуальні компоненти для наглядного представлення процесів та даних.

5. Міні-ігри: Дизайн міні-ігор повинен бути привабливим та захоплюючим. Візуальні ефекти, анімація та зручність керування можуть покращити користувацький досвід.

6. База даних: Система повинна мати добре структуровану базу даних для зберігання всієї інформації, отриманої під час взаємодії з користувачами. Це може включати таблиці для зберігання фінансових даних, даних про проекти, персоналу та інших важливих параметрів.

В цілому, дизайн та структура системи повинні забезпечувати зручну навігацію, логічний порядок дій та зручність використання для користувачів. Це допоможе їм ефективно взаємодіяти з системою та отримати бажані результати.

2.3 Вимоги до функціонування системи

2.3.1 Потреби користувача

Потреби користувачів навчально-інформаційної системи-симулятора діяльності фірми можуть бути різноманітними і охоплювати такі аспекти:

1. Навчання та освоєння навичок: Користувачі можуть мати потребу в набутті навичок управління фірмою, фінансового планування, інвестування та процесів прийняття рішень. Симулятор дозволяє їм практикувати та експериментувати в безпечному віртуальному середовищі.

2. Розвага та зацікавлення: Використання міні-ігор та елементів гри може задовольняти потребу користувачів в розвагах та відпочинку. Це може зробити процес навчання та експериментування більш привабливим і цікавим.

3. Практичне застосування знань: Користувачі можуть мати потребу в застосуванні набутих знань у реальному житті. Вони можуть бажати перевірити свої навички та здійснити певні дії, такі як взяття позики або інвестування в акції, щоб побачити результати своїх рішень.

4. Отримання повного образу фірми: Користувачі можуть бажати отримати повний образ процесів та взаємозв'язків у діяльності фірми. Вони можуть бути зацікавлені в розумінні різних ролей та відділів у фірмі, а також взаємодії між ними.

5. Оцінка результатів: Користувачі можуть мати потребу в оцінці своїх досягнень та результатів. Система може надавати звіти, статистику та оцінки їх продуктивності, фінансового стану та інших показників, що допоможе їм оцінити свої успіхи та вдосконалити свої навички.

2.3.2 Функціональні вимоги

Основні функціональні вимоги до навчально-інформаційної системи-симулятора діяльності фірми включають:

1. Головне меню: В системі має бути присутнє головне меню, яке надає користувачам доступ до основних функцій та модулів системи.

2. Симулятор фінансових операцій: Користувачі можуть взяти позику в банку під відсоток та вкладати кошти в акції для отримання доходу.

3. Модуль проектів: Користувачі можуть створювати нові проекти, вводити їх назви та отримувати інформацію про суму, яку їм заплатять за проект.

4. Модуль вибору розробників: Користувачі можуть обирати програміста з певною зарплатою та займатися міні-грою для розрахунку витрат на програміста.

5. Модуль вибору дизайнерів: Користувачі можуть обирати дизайнера з відповідною зарплатнею та займатися міні-грою для розрахунку витрат на дизайнера.

6. Модуль вибору тестувальників: Користувачі можуть обирати тестувальника та займатися міні-грою для розрахунку витрат на тестувальника.

7. Запис даних до бази даних: Всі дані про фінансові операції, проекти та витрати зберігаються в базі даних для подальшого аналізу та оцінки.

8. Звітність: Система надає користувачам звіти про отримані платежі за проект, суму витрат на розробників, дизайнерів та тестувальників, а також залишок коштів.

Ці функціональні вимоги забезпечують користувачам можливість взаємодії з системою, навчання, експериментування та оцінки результатів своєї діяльності.

2.3.3 Системні вимоги

Даний розділ визначає, розподіляє та вказує на системні вимоги, визначені розробником. Їх перелік наведений в таблиці А.1.

Таблиця А.1 – Системні вимоги

ID	Системні вимоги	Пріоритет	Опис
SR-01	Наявність головної форми	M	Головна форма з якої ми можемо перейти на сторінку відображення фінансових операцій, або почати створення нового проекту
SR-02	Наявність HTML сторінки	S	HTML сторінка яка надає можливість ознайомитися з роботою фондового ринку
SR-03	Наявність форм вибору виконавців	M	Форми вибору програміста, дизайнера та тестувальника
SR-04	Наявність форм міні ігор	S	Форми міні ігор, завдяки яким розраховується кількість днів та коштів витрачених на проект
SR-05	База даних з інформацією	M	База даних в яку записується вся інформація про проекти, виконавці, їх зарплатня, кість днів витрачених на розробку, сума витрачених коштів, та інше

Умовні позначення в таблиці А.1:

Must have (M) – вимоги, які повинні бути реалізовані в системі;

Should have (S) – вимоги, які мають бути виконані, але вони можуть почекати своєї черги;

2.4 Вимоги до видів забезпечення

2.4.1 Вимоги до інформаційного забезпечення

Розробка навчально інформаційної системи симулятора діяльності фірми відбувається з використанням:

– C#

- HTML 5
- CSS
- JavaScript
- База даних Access

2.4.2 Вимоги до лінгвістичного забезпечення

Система має бути на українській мові, але допустимі і англійські елементи

2.4.3 Вимоги до програмного забезпечення

Програмне забезпечення повинно задовольняти наступним вимогам

- Microsoft Visual Studio 2022
- Microsoft Studio Code 1.79.0
- Браузер Firefox 72.0.2

3. Склад і зміст робіт зі створення інформаційної системи

Докладний опис етапів роботи зі створення навчально інформаційної системи-симулятора фірми наведено в таблиці А.2.

Таблиці А.2 – Етапи створення інформаційної системи

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Аналіз методів реалізації	4 день
2	Створення функціоналу	40 днів
3	Створення додаткових форм	25 днів
4	Налаштування дизайну	16 днів
5	Тестування програми	2 дні
6	Допрацювання	4 дні
	Загальна тривалість робіт	91 день

4. Вимоги до складу й змісту робіт із введення інформаційної системи в експлуатацію

Вимоги до складу й змісту робіт із введення інформаційної системи в експлуатацію можуть включати наступне:

1. Підготовка середовища: Налаштування необхідного апаратного та програмного забезпечення для роботи системи. Це може включати встановлення серверів, баз даних, мережевого з'єднання та інших компонентів.

2. Тестування: Проведення різних видів тестів, таких як модульні тести, інтеграційні тести та системні тести, для перевірки працездатності системи та виявлення помилок та проблем.

3. Завантаження даних: Перенесення наявних даних (якщо такі є) у систему. Це може включати імпорт даних з інших джерел або вручне введення даних у систему.

4. Навчання користувачів: Проведення навчання та підготовки користувачів до роботи з новою системою. Це може включати навчання з використання системи, демонстрацію функціональності та надання пояснень.

5. Документація: Розробка необхідної документації, такої як посібник користувача, технічна документація, інструкції з установки та налаштування. Це допомагає користувачам отримати доступ до необхідної інформації та забезпечує належну підтримку системи.

6. Перехід до експлуатації: Планування та проведення переходу системи у робочий режим. Це може включати переключення на живлення системи, введення в роботу реальних користувачів та забезпечення необхідної технічної та функціональної підтримки.

Ці вимоги до складу й змісту робіт допомагають забезпечити успішне введення інформаційної системи в експлуатацію та готовність користувачів до ефективного використання системи.

Лістинг програмного коду

Код форми main_form:

```
public partial class main_form : Form
{
    public main_form()
    {
        InitializeComponent();
    }
    //новий проект
    private void button1_Click(object sender, EventArgs e)
    {
        project_name nameForm = new project_name();
        nameForm.Show();
        this.Hide();
    }
    //перегляд виконаних проектів
    private void button2_Click(object sender, EventArgs e)
    {
        project projectForm = new project();
        projectForm.Show();
        this.Hide();
    }
    // Закриття головної форми
    private void button3_Click(object sender, EventArgs e)
    {
        Form mainForm = this.FindForm();
        mainForm.Close();
        Application.Exit();
    }
}
```

```

private void main_form_FormClosed(object sender, FormClosedEventArgs e)
{
    Application.Exit();
}
private void button4_Click(object sender, EventArgs e)
{
    Process.Start("https://www.google.com/search?client=firefox-b-
d&q=%D0%B3%D1%83%D0%B3%D0%BB");
}
}

```

Код форми project_name:

```

public partial class project_name : Form
{
    //рядок підключення бази даних
    private OleDbConnection connection = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=Project.mdb;Persist Security Info=False;");
    public project_name()
    {
        InitializeComponent();
    }
    private void button1_Click(object sender, EventArgs e)
    {
        // Створення нового запису в базі даних та перехід до форми programmer
        connection.Open();
        OleDbCommand cmd = new OleDbCommand("SELECT COUNT(*) FROM
project", connection);
        int count = (int)cmd.ExecuteScalar();
        int w_id = count + 1;
        string w_name = comboBox1.Text;
    }
}

```

```

int colonIndex = w_name.IndexOf(':');
if (colonIndex > 0)
{
    w_name = w_name.Substring(0, colonIndex).Trim();
}

decimal projectCost = GetProjectCost(w_name); // Отримуємо вартість
проекту за його назвою
// Оновлення значення в лейблі
label3.Text = projectCost.ToString();
cmd.CommandText = "INSERT INTO project (w_id, w_name, w_price)
VALUES (@w_id, @w_name, @w_price)";
cmd.Parameters.AddWithValue("@w_id", w_id);
cmd.Parameters.AddWithValue("@w_name", w_name);
cmd.Parameters.AddWithValue("@w_price", projectCost);
cmd.ExecuteNonQuery();
connection.Close();
Choice choiseForm = new Choice(w_id);
choiseForm.Show();
this.Hide();
}

private void project_name_FormClosed(object sender, FormClosedEventArgs e)
{
    Application.Exit();
}

private void comboBox1_SelectedValueChanged(object sender, EventArgs e)
{
    string selectedProject = comboBox1.SelectedItem.ToString();
    decimal projectCost = GetProjectCost(selectedProject);
    label3.Text = projectCost.ToString();
}

```

```
private decimal GetProjectCost(string projectName)
{
    decimal projectCost = 0;
    // Отримати вартість проекту за його назвою
    if (projectName == "CloudSolutions")
    {
        projectCost = 40000;
    }
    else if (projectName == "AIgenius")
    {
        projectCost = 55000;
    }
    else if (projectName == "InnoTechLab")
    {
        projectCost = 46000;
    }
    else if (projectName == "VirtualRealityX")
    {
        projectCost = 70000;
    }
    else if (projectName == "SmartTech Solutions")
    {
        projectCost = 95000;
    }
    else if (projectName == "CodeGenius")
    {
        projectCost = 99000;
    }
    else if (projectName == "TechConnect")
    {
```

```

        projectCost = 83000;
    }
    return projectCost;
}
}

```

Код форми Choise:

```

public partial class Choice : Form
{
    private int w_id;
    public Choice(int w_id)
    {
        InitializeComponent();
        this.w_id = w_id;
    }
    private void button7_Click(object sender, EventArgs e)
    {
        // Закриття форми
        Form mainForm = this.FindForm();
        mainForm.Close();
        Application.Exit();
    }
    private void button8_Click(object sender, EventArgs e)
    {
        // Головне меню
        main_form mainForm = new main_form();
        mainForm.Show();
        this.Hide();
    }
    private void button3_Click(object sender, EventArgs e)
    {

```

```

        MessageBox.Show("Вірно, першим етапом є програмування проекту!",
"Молодець!");
        this.Close();
        programmer programmerForm = new programmer(w_id);
        programmerForm.Show();
    }
    private void button1_Click(object sender, EventArgs e)
    {
        MessageBox.Show("Цей етап буде пізніше!", "Помилка!");
    }
    private void button4_Click(object sender, EventArgs e)
    {
        MessageBox.Show("Цей етап буде пізніше!", "Помилка!");
    }
    private void button2_Click(object sender, EventArgs e)
    {
        MessageBox.Show("Цей етап не відноситься до нашого проекту!",
"Помилка!");
    }
    private void button5_Click(object sender, EventArgs e)
    {
        MessageBox.Show("Цей етап можна було б вважати вірним, але він не
відноситься до розробки проекту!", "Помилка!");
    }
}

```

Код форми programmer:

```

public partial class programmer : Form
{
    private int w_id;

```

```

private OleDbConnection connection = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=Project.mdb;Persist Security Info=False;");

public programmer(int w_id)
{
    InitializeComponent();
    this.w_id = w_id;
}

private void button1_Click(object sender, EventArgs e)
{
    // Додавання даних про програміста та його зарплатні до запису в базі
даних
    string programmer = "";
    int salary = 0;
    if (radioButton1.Checked)
    {
        programmer = radioButton1.Text;
        salary = 3000;
    }
    else if (radioButton2.Checked)
    {
        programmer = radioButton2.Text;
        salary = 2700;
    }
    else if (radioButton3.Checked)
    {
        programmer = radioButton3.Text;
        salary = 2900;
    }
    // Оновлюємо запис в базі даних

```

```

        string connectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=Project.mdb;";

        using (OleDbConnection connection = new
OleDbConnection(connectionString))
        {
            connection.Open();

            OleDbCommand command = new OleDbCommand("UPDATE project SET
w_programmer = @programmer, w_programmer_salary = @salary WHERE w_id =
(SELECT MAX(w_id) FROM project)", connection);

            command.Parameters.AddWithValue("@programmer", programmer);
            command.Parameters.AddWithValue("@salary", salary);
            command.ExecuteNonQuery();
        }

        MessageBox.Show("Гра П'ятнашки, вам потрібно розставити всі цифри в
вірному порядку, в цей час будуть рахуватися дні виконання (10 секунд
вважається як 1 день) Бажаю успіху!", "Інструкція");

        programmer_game PGForm = new programmer_game(w_id, salary);
        PGForm.Show();
        this.Hide();
    }
}

```

Код форми programmer_game:

```

public partial class programmer_game : Form
{
    private int salary;
    private int w_id;
    private int BOARD_SIZE;
    private int SQUARE_SIZE;

    private const int BOARD_OFFSET_X = 15; // Відступ від лівого краю вікна
(пікселів)

```

```

private const int BOARD_OFFSET_Y = 15; // Відступ від верхнього краю вікна
(пікселів)

private Label[,] squares; //= new Label[BOARD_SIZE, BOARD_SIZE];
private int emptyRow;
private int emptyCol;
private int days_p = 0; // зберігає кількість днів, які вже минули
private int budget_p = 0; //зберігає бюджет використаний на программіста
private const int SHUFFLE_MOVES = 100; // Кількість переміщуваль клітинок
public programmer_game(int w_id, int salary)
{
    InitializeComponent();
    this.salary = salary;
    this.w_id = w_id;
    // Визначаємо кількість квадратів і їх розмір
    if (salary == 3000)
    {
        BOARD_SIZE = 3;
        SQUARE_SIZE = 80;
    }
    else if (salary == 2700)
    {
        BOARD_SIZE = 5;
        SQUARE_SIZE = 57;
    }
    else if (salary == 2900)
    {
        BOARD_SIZE = 4;
        SQUARE_SIZE = 70;
    }
    // Розміщуємо квадрати на формі

```

```

squares = new Label[BOARD_SIZE, BOARD_SIZE];
for (int row = 0; row < BOARD_SIZE; row++)
{
    for (int col = 0; col < BOARD_SIZE; col++)
    {
        Label square = new Label();
        square.Size = new Size(SQUARE_SIZE, SQUARE_SIZE);
        square.BorderStyle = BorderStyle.FixedSingle;
        square.TextAlign = ContentAlignment.MiddleCenter;
        square.Font = new Font("Arial", 24, FontStyle.Bold);
        square.Text = $"{row * BOARD_SIZE + col + 1}";
        square.Location = new Point(col * SQUARE_SIZE +
BOARD_OFFSET_X, row * SQUARE_SIZE + BOARD_OFFSET_Y);
        square.Click += Square_Click;
        squares[row, col] = square;
        this.Controls.Add(square);
    }
}

// Випадково перемішуємо квадрати
Shuffle();
}

private void Square_Click(object sender, EventArgs e)
{
    Label clickedSquare = sender as Label;

    // Отримуємо рядок і стовпець клікнутого квадрата
    int clickedRow = -1;
    int clickedCol = -1;
    for (int row = 0; row < BOARD_SIZE; row++)
    {

```

```

for (int col = 0; col < BOARD_SIZE; col++)
{
    if (squares[row, col] == clickedSquare)
    {
        clickedRow = row;
        clickedCol = col;
        break;
    }
}
if (clickedRow != -1 && clickedCol != -1)
{
    break;
}
}

// Перевіряємо чи клікнутий квадрат знаходиться поруч з порожнім
if ((clickedRow == emptyRow && Math.Abs(clickedCol - emptyCol) == 1) ||
    (clickedCol == emptyCol && Math.Abs(clickedRow - emptyRow) == 1))
{
    // Міняємо місцями клітинки
    squares[emptyRow, emptyCol].Text = clickedSquare.Text;
    clickedSquare.Text = ""; // Оновлюємо позиції порожньої клітинки
    emptyRow = clickedRow;
    emptyCol = clickedCol;
    // Перевіряємо, чи гра завершена
    if (IsGameComplete())
    {
        budget_p = salary * days_p;
        string connectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=Project.mdb;";

```

```

        using (OleDbConnection connection = new
OleDbConnection(connectionString))
        {
            connection.Open();

            OleDbCommand command = new OleDbCommand("UPDATE project
SET w_days = w_days + @days_p, w_budget = w_budget + @budget_p WHERE w_id
= (SELECT MAX(w_id) FROM project)", connection);

            command.Parameters.AddWithValue("@days_d", days_p);
            command.Parameters.AddWithValue("@budget_d", budget_p);
            command.ExecuteNonQuery();
        }

        // MessageBox.Show("Ви успішно завершили програмування проекту,
витративши на нього {days_p} днів, відповідно {budget_p} грн!");

        MessageBox.Show("Ви успішно завершили програмування проекту,
витративши на нього " + days_p + " дні і відповідно " + budget_p + " грн",
"Виконано!");

        ReturnToMainForm();
    }
}

private void Shuffle()
{
    Random random = new Random();
    // Встановлюємо початкову порожню клітинку
    int lastRow = BOARD_SIZE - 1;
    int lastCol = BOARD_SIZE - 1;
    squares[lastRow, lastCol].Text = "";
    emptyRow = lastRow;
    emptyCol = lastCol;
}

```

```

for (int i = 0; i < 100; i++)
{
    int randomRow, randomCol;
    do
    {
        randomRow = random.Next(BOARD_SIZE);
        randomCol = random.Next(BOARD_SIZE);
    }
    while ((Math.Abs(randomRow - lastRow) == 1 && randomCol == lastCol) ||
           (Math.Abs(randomCol - lastCol) == 1 && randomRow == lastRow));
    Swap(randomRow, randomCol, lastRow, lastCol);
    lastRow = randomRow;
    lastCol = randomCol;
}

emptyRow = lastRow;
emptyCol = lastCol;
UpdateBoard();
}

private void Swap(int row1, int col1, int row2, int col2)
{
    string tempText = squares[row1, col1].Text;
    squares[row1, col1].Text = squares[row2, col2].Text;
    squares[row2, col2].Text = tempText;
}

private void UpdateBoard()
{
    for (int row = 0; row < BOARD_SIZE; row++)
    {
        for (int col = 0; col < BOARD_SIZE; col++)

```

```

    {
        squares[row, col].Location = new Point(col * SQUARE_SIZE +
BOARD_OFFSET_X, row * SQUARE_SIZE + BOARD_OFFSET_Y);
    }
}
}
private bool IsGameComplete()
{
    int count = 1;
    for (int row = 0; row < BOARD_SIZE; row++)
    {
        for (int col = 0; col < BOARD_SIZE; col++)
        {
            if (row == emptyRow && col == emptyCol)
            {
                // Пропускаємо порожню клітинку
                continue;
            }
            int expectedValue = (row * BOARD_SIZE + col + 1) % (BOARD_SIZE *
BOARD_SIZE); // Очікуване значення на поточній позиції
            if (squares[row, col].Text != $"{expectedValue}")
            {
                return false;
            }
            count++;
        }
    }
    // Перевірка, чи порожній квадрат знаходиться на останній позиції
    if (emptyRow == BOARD_SIZE - 1 && emptyCol == BOARD_SIZE - 1)
    {

```

```

        return true;
    }
    return false;
}

private void ReturnToMainForm()
{
    // Закриваємо поточну форму та створюємо нову головну форму
    designer DForm = new designer(w_id);
    DForm.Show();
    this.Hide();
}

private void timer3_Tick(object sender, EventArgs e)
{
    days_p++; // збільшуємо лічильник днів

    label1.Text = days_p.ToString() + " днів"; // оновлюємо текст Label на формі
}

private void button20_Click(object sender, EventArgs e)
{
    Random random = new Random();
    int lastRow = emptyRow;
    int lastCol = emptyCol;
    for (int i = 0; i < 100; i++)
    {
        int randomRow, randomCol;
        do
        {
            randomRow = random.Next(BOARD_SIZE);
            randomCol = random.Next(BOARD_SIZE);
        }
    }
}

```

```

while ((Math.Abs(randomRow - lastRow) == 1 && randomCol == lastCol) ||
      (Math.Abs(randomCol - lastCol) == 1 && randomRow == lastRow));
Swap(randomRow, randomCol, lastRow, lastCol);
lastRow = randomRow;
lastCol = randomCol;
}
emptyRow = lastRow;
emptyCol = lastCol;
UpdateBoard();
}
}

```

Код формы designer:

```

public partial class designer : Form
{
    private int w_id;
    private OleDbConnection connection = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=Project.mdb;Persist Security Info=False;");
    public designer(int w_id)
    {
        InitializeComponent();
        this.w_id = w_id;
    }
    private void button1_Click(object sender, EventArgs e)
    {
        string designer = "";
        int salary = 0;
        if (radioButton1.Checked)
        {
            designer = radioButton1.Text;

```

```

        salary = 2500;
    }
    else if (radioButton2.Checked)
    {
        designer = radioButton2.Text;
        salary = 2200;
    }
    else if (radioButton3.Checked)
    {
        designer = radioButton3.Text;
        salary = 2600;
    }
    // Оновлюємо запис в базі даних
    string connectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=Project.mdb;";
    using (OleDbConnection connection = new
OleDbConnection(connectionString))
    {
        connection.Open();
        OleDbCommand command = new OleDbCommand("UPDATE project SET
w_designer = @designer, w_designer_salary = @salary WHERE w_id = (SELECT
MAX(w_id) FROM project)", connection);
        command.Parameters.AddWithValue("@designer", designer);
        command.Parameters.AddWithValue("@salary", salary);
        command.ExecuteNonQuery();
    }
    MessageBox.Show("Гра клон OSU, вам потрібно контуром круга ловити
сам круг, потрібно набрати 20 очок, в цей час будуть рахуватися дні виконання (5
секунд вважається як 1 день) Бажаю успіху!", "Інструкція");
    designer_game DGForm = new designer_game(w_id, salary);

```

```

        DGForm.Show();
        this.Close();
    }
    private void designer_FormClosed(object sender, FormClosedEventArgs e)
    {
        Application.Exit();
    }
}

```

Код формы designer_game:

```

public partial class designer_game : Form
{
    public Bitmap HandlerTexture = Resource1.Handler;
    public Bitmap TargetTexture = Resource1.Target;
    private Point _targetPosition = new Point(300, 300);
    private Point _direction = Point.Empty;
    private int _score = 0;
    private int w_id;
    private int salary;
    private int days_d = 0;
    private int budget_d = 0;
    public designer_game(int w_id, int salary)
    {
        InitializeComponent();
        this.w_id = w_id;
        this.salary = salary;
        SetStyle(ControlStyles.OptimizedDoubleBuffer |
            ControlStyles.AllPaintingInWmPaint |
            ControlStyles.UserPaint, true);

        UpdateStyles();
    }
}

```

```

    }
    private void timer1_Tick(object sender, EventArgs e)
    {
        Invalidate();
        // Refresh();
    }
    private void timer2_Tick(object sender, EventArgs e)
    {
        Random r = new Random();
        timer2.Interval = r.Next(500, 6000);
        _direction.X = r.Next(1, 2);
        _direction.Y = r.Next(1, 2);
    }
    private void designer_game_Paint(object sender, PaintEventArgs e)
    {
        Graphics g = e.Graphics;
        var localPosition = this.PointToClient(Cursor.Position);
        if (_targetPosition.X < 0 || _targetPosition.X > 500)
        {
            _direction.X *= -1;
        }
        if (_targetPosition.Y < 0 || _targetPosition.Y > 350)
        {
            _direction.Y *= -1;
        }
        Point between = new Point(localPosition.X - _targetPosition.X, localPosition.Y
- _targetPosition.Y);
        float distance = (float)Math.Sqrt((between.X * between.X) + (between.Y *
between.Y));
        if (salary == 2500)

```

```

{
    _targetPosition.X += _direction.X * 6;
    _targetPosition.Y += _direction.Y * 6;
    if (distance < 10)
    {
        AddScore(1);
    }
}
else if (salary == 2200)
{
    _targetPosition.X += _direction.X * 4;
    _targetPosition.Y += _direction.Y * 4;
    if (distance < 5)
    {
        AddScore(1);
    }
}
else if (salary == 2600)
{
    _targetPosition.X += _direction.X * 10;
    _targetPosition.Y += _direction.Y * 10;
    if (distance < 15)
    {
        AddScore(1);
    }
}

var handlerRect = new Rectangle(localPosition.X - 50, localPosition.Y - 50,
100, 100);

var targetReact = new Rectangle(_targetPosition.X - 50, _targetPosition.Y - 50,
100, 100);

```

```

g.DrawImage(TargetTexture, handlerRect);
g.DrawImage(HandlerTexture, targetReact);
if (_score == 20)
{
    budget_d = salary * days_d;
    string connectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=Project.mdb;";
    using (OleDbConnection connection = new
OleDbConnection(connectionString))
    {
        connection.Open();
        OleDbCommand command = new OleDbCommand("UPDATE project
SET w_days = w_days + @days_d, w_budget = w_budget + @budget_d WHERE w_id
= (SELECT MAX(w_id) FROM project)", connection);
        command.Parameters.AddWithValue("@days_d", days_d);
        command.Parameters.AddWithValue("@budget_d", budget_d);
        command.ExecuteNonQuery();
    }

    this.Close();
    //MessageBox.Show("Ви завершили дизайн проєкту,тепер перейдемо до
відділу тестування!", "Виконано");
    MessageBox.Show("Ви успішно завершили програмування проєкту,
витративши на нього " + days_d + " дні і відповідно " + budget_d + " грн",
"Виконано!");
    testing testingForm = new testing(w_id);
    testingForm.Show();
}
}

```

```
private void AddScore(int score)
{
    _score += score;
    scorelabel.Text = "Зароблено: " + _score.ToString() + " очок"; ;
}
```

```
private void timer3_Tick(object sender, EventArgs e)
```

```
{
    days_d++; // збільшуємо лічильник днів
    label1.Text = "Витрачено: " + days_d.ToString() + " днів"; // оновлюємо
```

текст Label на формі

```
}
private void designer_game_FormClosing(object sender, FormClosingEventArgs
```

e)

```
{
    timer1.Stop();
    timer2.Stop();
    timer3.Stop();
}
```

```
}
```

Код форми testing:

```
public partial class testing : Form
```

```
{
```

```
    private int w_id;
```

```
    private OleDbConnection connection = new
```

```
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data
```

```
Source=Project.mdb;Persist Security Info=False;");
```

```
    public testing(int w_id)
```

```
{
```

```
        InitializeComponent();
```

```
        this.w_id = w_id;
```

```

    }
    private void button1_Click(object sender, EventArgs e)
    {
        // Додавання даних про програміста та його зарплатні до запису в базі
даних
        string tester = "";
        int salary = 0;
        if (radioButton1.Checked)
        {
            tester = radioButton1.Text;
            salary = 2300;
        }
        else if (radioButton2.Checked)
        {
            tester = radioButton2.Text;
            salary = 1800;
        }
        else if (radioButton3.Checked)
        {
            tester = radioButton3.Text;
            salary = 2100;
        }
        // Оновлюємо запис в базі даних
        string connectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=Project.mdb;";
        using (OleDbConnection connection = new
OleDbConnection(connectionString))
        {
            connection.Open();

```

```

OleDbCommand command = new OleDbCommand("UPDATE project SET
w_tester = @programmer, w_tester_salary = @salary WHERE w_id = (SELECT
MAX(w_id) FROM project)", connection);
    command.Parameters.AddWithValue("@tester", tester);
    command.Parameters.AddWithValue("@salary", salary);
    command.ExecuteNonQuery();
}

```

MessageBox.Show("Гра Хрестики нолики, вам потрібно набрати 10 очок, в цей час будуть рахуватися дні виконання (5 секунд вважається як 1 день) Бажаю успіху!", "Інструкція");

```

    testing_game TGForm = new testing_game(w_id, salary);
    TGForm.Show();
    this.Hide();
}
}

```

Код форми testing_game:

```

public partial class testing_game : Form
{
    private int w_id;
    private int salary;
    private int days_t = 0;
    private int budget_t = 0;
    private int _score = 0;
    private int money_left = 0;
    public testing_game(int w_id, int salary)
    {
        InitializeComponent();
        this.w_id = w_id;
        this.salary = salary;
        this.Height = 420; // Встановлення висоти форми
    }
}

```

```

this.Width = 350; // Встановлення ширини форми
player = 1; // Змінна, що визначає гравця, який ходить (1 або 0)
int buttonSize = GetButtonSize(); // Отримання розміру кнопок
int buttonCount = GetButtonCount(); // Отримання кількості кнопок
buttons = new Button[buttonCount, buttonCount]; // Оновлення розміру масиву
кнопок
setButtons(buttonSize, buttonCount); // Метод для розміщення кнопок на формі
}

private Button[,] buttons; // Масив кнопок розміром 3x3
private int player; // Змінна, що визначає гравця, який ходить (1 або 0)
private int GetButtonSize()
{
    if (salary == 2300)
    {
        return 60; // Розмір кнопок для salary = 2300
    }
    else if (salary == 1800)
    {
        return 100; // Розмір кнопок для salary = 1800
    }
    else
    {
        return 70; // Розмір кнопок для salary = 2100
    }
}

private int GetButtonCount()
{
    if (salary == 2300)
    {

```

```

        return 5; // Кількість кнопок для salary = 2300
    }
    else if (salary == 1800)
    {
        return 3; // Кількість кнопок для salary = 1800
    }
    else
    {
        return 4; // Кількість кнопок для salary = 2100
    }
}

private void setButtons(int buttonSize, int buttonCount)
{
    for (int i = 0; i < buttonCount; i++)
    {
        for (int j = 0; j < buttonCount; j++)
        {
            buttons[i, j] = new Button();
            buttons[i, j].Size = new Size(buttonSize, buttonSize); // Встановлення
розміру кнопки
            buttons[i, j].Location = new Point(2 + (buttonSize + 6) * j, 2 + (buttonSize +
6) * i); // Встановлення розташування кнопки на формі
            buttons[i, j].Click += button1_Click; // Підписка на подію Click для кнопки
            buttons[i, j].Font = new Font(new FontFamily("Microsoft Sans Serif"),
buttonSize - 30); // Встановлення шрифту кнопки
            buttons[i, j].Text = ""; // Початковий текст кнопки
            this.Controls.Add(buttons[i, j]); // Додавання кнопки до елементів
управління форми
        }
    }
}

```

```

    }
    private void button1_Click(object sender, EventArgs e)
    {
        Button clickedButton = (Button)sender;
        // Хід гравця
        if (player == 1)
        {
            clickedButton.Text = "x"; // Встановлення тексту "x" для кнопки
            player = 0; // Змінна, що визначає гравця, який ходить (змінюється на 0)
            clickedButton.Enabled = false; // Вимкнення можливості натискати кнопку
            checkWin(); // Перевірка наявності переможця та завершення гри
            if (!CheckGameOver() && player == 0)
                MakeComputerMove(); // Хід комп'ютера
        }
    }
    private void MakeComputerMove()
    {
        // Штучний інтелект для ходу комп'ютера
        // Наприклад, випадковий вибір вільної кнопки
        List<Button> emptyButtons = GetEmptyButtons(); // Отримання списку
        вільних кнопок
        if (emptyButtons.Count > 0)
        {
            Random random = new Random();
            int randomIndex = random.Next(0, emptyButtons.Count); //
            Випадковий вибір індексу кнопки зі списку вільних кнопок
            Button selectedButton = emptyButtons[randomIndex];
            selectedButton.Text = "o"; // Встановлення тексту "o" для вибраної кнопки
            selectedButton.Enabled = false; // Вимкнення можливості натискати кнопку
            player = 1; // Зміна гравця на 1
        }
    }

```

```

        // label1.Text = "Текущий ход: Игрок 1";
        checkWin(); // Перевірка наявності переможця та завершення гри
    }
}

private List<Button> GetEmptyButtons()
{
    List<Button> emptyButtons = new List<Button>();
    foreach (Button button in buttons)
    {
        if (string.IsNullOrEmpty(button.Text))
        {
            emptyButtons.Add(button); // Додавання вільної кнопки до списку
ВІЛЬНИХ КНОПОК
        }
    }
    return emptyButtons; // Повернення списку вільних кнопок
}

private bool CheckGameOver()
{
    foreach (Button button in buttons)
    {
        if (string.IsNullOrEmpty(button.Text))
        {
            return false; // Гра ще не завершена (є ще вільні кнопки)
        }
    }
    ResetGame(); // Скидання гри до початкових налаштувань
    return true;
}

private void checkWin()

```

```
{
    int buttonCount = GetButtonCount();
    int rowCount = GetRowCount();
    int colCount = GetColumnCount();
    // Перевірка по горизонталі
    for (int row = 0; row < rowCount; row++)
    {
        for (int col = 0; col <= colCount - 3; col++)
        {
            if (CheckLine(row, col, row, col + 1, row, col + 2))
            {
                DeclareWinner();
                return;
            }
        }
    }
    // Перевірка по вертикалі
    for (int col = 0; col < colCount; col++)
    {
        for (int row = 0; row <= rowCount - 3; row++)
        {
            if (CheckLine(row, col, row + 1, col, row + 2, col))
            {
                DeclareWinner();
                return;
            }
        }
    }
    // Перевірка по головній діагоналі
    for (int row = 0; row <= rowCount - 3; row++)
```

```

{
    for (int col = 0; col <= colCount - 3; col++)
    {
        if (CheckLine(row, col, row + 1, col + 1, row + 2, col + 2))
        {
            DeclareWinner();
            return;
        }
    }
}

// Перевірка по побічній діагоналі
for (int row = 0; row <= rowCount - 3; row++)
{
    for (int col = colCount - 1; col >= 2; col--)
    {
        if (CheckLine(row, col, row + 1, col - 1, row + 2, col - 2))
        {
            DeclareWinner();
            return;
        }
    }
}

// Перевірка на нічию
if (CheckGameOver())
{
    MessageBox.Show("Гра закінчилася нічиєю!");
    ResetGame();
}
}

private int GetRowCount()

```

```

{
    return buttons.GetLength(0);
}
private int GetColumnCount()
{
    return buttons.GetLength(1);
}
private bool CheckLine(int row1, int col1, int row2, int col2, int row3, int col3)
{
    string text1 = buttons[row1, col1].Text;
    string text2 = buttons[row2, col2].Text;
    string text3 = buttons[row3, col3].Text;
    return !string.IsNullOrEmpty(text1) && text1 == text2 && text2 == text3;
}
private void DeclareWinner()
{
    if (player == 0)
    {
        AddScore(1);
        ResetGame();// Сдання гри до початкових налаштувань
    }
    else if (player == 1)
    {
        MessageBox.Show("Спробуйте ще раз!");
        ResetGame();// Сдання гри до початкових налаштувань
    }
}
private void ResetGame()
{
    foreach (Button button in buttons)

```

```

{
    button.Text = ""; // Очищення тексту кнопки
    button.Enabled = true; // Включення можливості натискати кнопку
}
player = 1;
if (_score == 10)
{
    decimal price;
    budget_t = days_t * salary;
    decimal money_left;

    string connectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=Project.mdb;";
    using (OleDbConnection connection = new OleDbConnection(connectionString))
    {
        connection.Open();
        // Отримати значення price з бази даних
        OleDbCommand priceCommand = new OleDbCommand("SELECT
w_price FROM project WHERE w_id = (SELECT TOP 1 w_id FROM project ORDER
BY w_id DESC)", connection);
        price = Convert.ToDecimal(priceCommand.ExecuteScalar());
        money_left = price - budget_t;
        OleDbCommand command = new OleDbCommand("UPDATE
project SET w_days = w_days + @days_t, w_budget = w_budget + @budget_t,
w_money_left = @money_left WHERE w_id = (SELECT TOP 1 w_id FROM project
ORDER BY w_id DESC)", connection);
        command.Parameters.AddWithValue("@days_t", days_t);
        command.Parameters.AddWithValue("@budget_t", budget_t);
        command.Parameters.AddWithValue("@money_left", money_left);
        command.ExecuteNonQuery();
        // Вибірка даних з полів w_days, w_budget та w_money_left

```

```

        OleDbCommand selectCommand = new OleDbCommand("SELECT
w_name, w_days, w_budget, w_money_left FROM project WHERE w_id = (SELECT
TOP 1 w_id FROM project ORDER BY w_id DESC)", connection);

        OleDbDataReader reader = selectCommand.ExecuteReader();
        if (reader.Read())
        {
            string w_name = reader.GetString(reader.GetOrdinal("w_name"));
            int w_days = Convert.ToInt32(reader["w_days"]);
            int w_budget = Convert.ToInt32(reader["w_budget"]);
            decimal w_money_left =
reader.GetDecimal(reader.GetOrdinal("w_money_left"));

            this.Close();

            MessageBox.Show("Ви успішно завершили тестування проекту,
витративши на нього " + days_t + " дні, та відповідно " + budget_t + " грн.",
"Виконано!");

            MessageBox.Show("Ви завершили виконання проекту " + w_name + "
витративши на нього " + w_days + " днів, та " + w_budget + " грн", "Проект
завершено");

        }

        reader.Close();

        report report= new report();
        report.Show();
    }
}

private void AddScore(int score)
{
    _score += score;
    scorelabel.Text = "Зароблено: \n" + _score.ToString() + " очок"; ;
}

```

```
private void timer3_Tick(object sender, EventArgs e)
{
    days_t++; // збільшуємо лічильник днів
    label1.Text = "Витрачено: \n" + days_t.ToString() + " днів"; //
```

оновлюємо текст Label на формі

```
    }
    private void testing_game_FormClosing(object sender,
FormClosingEventArgs e)
    {
        timer3.Stop();
    }
}
```

Код форми report:

```
public partial class report : Form
{
    private OleDbConnection connection = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=Project.mdb;Persist Security Info=False;");
    private decimal price;
    private decimal budget;

    public report()
    {
        InitializeComponent();
    }
    private void investmens_Load(object sender, EventArgs e)
    {
        LoadDataFromDatabase();
    }
}
```

```

private void LoadDataFromDatabase()
{
    try
    {
        connection.Open();
        OleDbCommand cmd = new OleDbCommand("SELECT w_price, w_budget,
w_money_left FROM project", connection);
        OleDbDataReader reader = cmd.ExecuteReader();
        if (reader.Read())
        {
            price = reader.GetDecimal(0);
            budget = reader.GetDecimal(1);
            decimal moneyLeft = reader.GetDecimal(2);

            label2.Text = price.ToString();
            label5.Text = budget.ToString();
            label8.Text = moneyLeft.ToString();
        }
        reader.Close();
    }
    catch (Exception ex)
    {
        // Обробка винятків
    }
    finally
    {
        connection.Close();
    }
}

private void UpdateMoneyLeft()

```

```

{
    try
    {
        decimal moneyLeft = budget - price;
        label8.Text = moneyLeft.ToString();

        connection.Open();
        OleDbCommand cmd = new OleDbCommand("UPDATE project SET
w_money_left = @moneyLeft", connection);
        cmd.Parameters.AddWithValue("@moneyLeft", moneyLeft);
        cmd.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        // Обробка винятків
    }
    finally
    {
        connection.Close();
    }
}

private void button3_Click(object sender, EventArgs e)
{
    Application.Exit();
}

private void button2_Click(object sender, EventArgs e)
{
    main_form MainForm = new main_form();
    MainForm.Show();
}

```

```

        this.Hide();
    }
    private void investmens_Shown(object sender, EventArgs e)
    {
        UpdateMoneyLeft();
    }
}

```

Код форми project:

```

public partial class project : Form
{
    public project()
    {
        InitializeComponent();
    }
}

```

```

private void project_Load(object sender, EventArgs e)
{
    // Підключення до бази даних Access
    string connectionString = @"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=Project.mdb";
    using (OleDbConnection connection = new
OleDbConnection(connectionString))
    {
        // Запит для отримання даних з таблиці
        string query = "SELECT w_name, w_programmer, w_designer, w_tester,
w_days, w_budget, w_price, w_money_left FROM project";
        // Створення команди
        OleDbCommand command = new OleDbCommand(query, connection);
        // Відкриття з'єднання з базою даних
    }
}

```

```
connection.Open();
// Виконання команди та отримання результату
OleDbDataReader reader = command.ExecuteReader();
// Очищення ListView
listView1.Items.Clear();
listView1.Columns.Clear();
// Налаштування властивості View на View.Details
listView1.View = View.Details;
// Додавання стовпців у ListView
listView1.Columns.Add("Назва", 110);
listView1.Columns.Add("Програміст", 80);
listView1.Columns.Add("Дизайнер", 80);
listView1.Columns.Add("Тестувальник", 80);
listView1.Columns.Add("дні", 50);
listView1.Columns.Add("Витрачено", 70);
listView1.Columns.Add("Ціна", 70);
listView1.Columns.Add("Залишок", 70);
// Додавання рядків до ListView
while (reader.Read())
{
    ListViewItem item = new ListViewItem(reader["w_name"].ToString());
    item.SubItems.Add(reader["w_programmer"].ToString());
    item.SubItems.Add(reader["w_designer"].ToString());
    item.SubItems.Add(reader["w_tester"].ToString());
    item.SubItems.Add(reader["w_days"].ToString());
    item.SubItems.Add(reader["w_budget"].ToString() + " грн");
    item.SubItems.Add(reader["w_price"].ToString() + " грн");
    item.SubItems.Add(reader["w_money_left"].ToString() + " грн");
    listView1.Items.Add(item);
}
```

```

        // Закриття з'єднання з базою даних
        connection.Close();
    }
}

private void project_FormClosed(object sender, FormClosedEventArgs e)
{
    Application.Exit();
}

private void button1_Click(object sender, EventArgs e)
{
    // Підключення до бази даних Access
    string connectionString = @"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=Project.mdb;";
    using (OleDbConnection connection = new
OleDbConnection(connectionString))
    {
        // Запит для видалення всіх записів з таблиці
        string query = "DELETE FROM project";
        // Створення команди
        OleDbCommand command = new OleDbCommand(query, connection);
        // Відкриття з'єднання з базою даних
        connection.Open();
        // Виконання команди
        command.ExecuteNonQuery();
        // Закриття з'єднання з базою даних
        connection.Close();
    }
    // Очищення ListBox
    listView1.Items.Clear();
}

```

```
private void button2_Click(object sender, EventArgs e)
{
    main_form mainForm = new main_form();
    mainForm.Show();
    this.Hide();
}
}
```

Головний файл стилів index.css:

```
p, h1, h3 {
    color: #e1e1e1;
}

body {
    overflow: scroll;
}

#terminal {
    background-color: #000000;
    width: 80vw;
    height: 10vh;
    color: #e1e1e1;
    margin: 2vh 2vw 2vh 2vw;
    border-radius: 25px;
    padding: 10px;
}

#terminal > p, #info > p {
    margin: 0.4vh;
    margin-left: 0.4vh;
```

```
    font-size: 1.3vh;  
}
```

```
#info {  
    position: absolute;  
    right: 2vw;  
    top: 2vh;  
    background-color: #000000;  
    width: 14vw;  
    height: 10vh;  
    color: #e1e1e1;  
    border-radius: 25px;  
    padding: 10px;  
    margin-left: 15px;  
}
```

```
#game, #startinfo {  
    height: 82vh;  
    width: 96vw;  
    margin: 2vh 2vw 2vh 2vw;  
    border-radius: 25px;  
    background-color: #050505;  
}
```

```
#tabs {  
    margin: auto;  
    display: grid;  
    grid-gap: 10px;  
    border-radius: 25px;  
    width: 98%;
```

```
}
```

```
#tabs > div:nth-child(1) {  
  border-radius: 25px 0 0 25px;  
}
```

```
#tabs > div:last-child {  
  border-radius: 0 25px 25px 0;  
}
```

```
#tabs > div:nth-child(even) {  
  background-color: #202020;  
}
```

```
#tabs > div {  
  color: #e1e1e1;  
  background-color: #101010;  
  grid-row-start: 1;  
  text-align: center;  
  font-size: 1.5em;  
}
```

```
#tabs > div:hover, button:hover, input:hover {  
  background-color: #494989;  
  box-shadow: inset 0 -3em 3em #202060, 0 0 0 2px #202060,  
    0.3em 0.3em 1em #202060;  
}
```

```
.title {  
  padding: 20px;
```

```
margin: 0;
text-align: center;
color: #e1e1e1;
}
```

```
.center {
  display: flex;
  align-items: center;
  justify-content: center;
  gap: 20px;
}
```

```
.menu {
  margin-top: 2em;
}
```

```
#stockscontainer {
  display: grid;
  grid-template-columns: repeat(3, minmax(0, 1fr));
  gap: 2vw 2vw;
  margin: auto;
  grid-auto-rows: 100px;
  padding: 20px;
}
```

```
#stockscontainer > div > div:nth-child(even) {
  background-color: #202020;
}
```

```
#stockscontainer > div > div {
```

```
background-color: #101010;
}
```

```
.stockdec {
  color: #ef1919;
}
```

```
.stockinc {
  color: #0d920d;
}
```

```
#loanbuttons {
  display: grid;
  width: 30vw;
  height: 60vh;
  gap: 2vw 2vw;
  margin: auto;
  grid-template-columns: repeat(2, minmax(0, 1fr));
}
```

```
#loanbuttons > button:nth-child(even) {
  grid-column-start: 2;
}
```

```
#loanbuttons > button {
  grid-column-start: 1;
  font-size: 2vh;
}
```

```
#bankruptbutton {
```

```
margin: 0;
width: 40vw;
height: 40vh;
font-size: 8vh;
}
```

```
button,
input {
  background-color: #303030;
  color: #a0a0a0;
  border-color: #000040;
}
```

```
body {
  background-color: #101010;
  overflow: hidden;
}
```

Блок у файлі index.html який відповідає відображенню акцій компаній:

```
<div style="display: none" class="menu">
  <h1 class="title">Інвестиції</h1>
  <div id="stockscontainer">
    <div id="lowrisk">
      <h1>Низький ризик</h1>
      <div class="stock">
        <h3>KERNEL – 0</h3>
        <p>За одну акцію: 10 грн <br> У власності: 0</p><input type="text"
id="stock1" placeholder="Кількість для покупки">
        <button
```

```
onclick="BuyStock(0,1,document.getElementById('stock1').value)">Покупка</button>
```

```
<button onclick="BuyStock(0,1,-
document.getElementById('stock1').value)">Продаж</button>
</div>
```

```
<div class="stock">
<h3>FERREXPO – 0</h3>
<p>За одну акцію: 100 грн <br> У власності: 0</p><input type="text"
id="stock2" placeholder="Кількість для покупки">
```

```
<button
onclick="BuyStock(0,2,document.getElementById('stock2').value)">Покупка</button>
```

```
<button onclick="BuyStock(0,2,-
document.getElementById('stock2').value)">Продаж</button>
</div>
```

```
<div class="stock">
<h3>AGROTON – 0</h3>
<p>За одну акцію: 1000 грн <br> У власності: 0</p><input type="text"
id="stock3" placeholder="Кількість для покупки">
```

```
<button
onclick="BuyStock(0,3,document.getElementById('stock3').value)">Покупка</button>
```

```
<button onclick="BuyStock(0,3,-
document.getElementById('stock3').value)">Продаж</button>
</div>
```

```
</div>
```

```
<div id="medrisk">
```

```
<h1>Середній ризик</h1>
```

```
<div class="stock">
```

```
<h3>MXP – 0</h3>
```

```

        <p>За одну акцію: 10 грн <br> У власності: 0</p><input type="text"
id="stock4" placeholder="Кількість для покупки">
        <button
onclick="BuyStock(1,1,document.getElementById('stock4').value)">Покупка</button
>
        <button onclick="BuyStock(1,1,-
document.getElementById('stock4').value)">Продажа</button>
    </div>
    <div class="stock">
        <h3>UKRNAFTA – 0</h3>
        <p>За одну акцію: 100 грн <br> У власності: 0</p><input type="text"
id="stock5" placeholder="Кількість для покупки">
        <button
onclick="BuyStock(1,2,document.getElementById('stock5').value)">Покупка</button
>
        <button onclick="BuyStock(1,2,-
document.getElementById('stock5').value)">Продажа</button>
    </div>
    <div class="stock">
        <h3>Tesla – 0</h3>
        <p>За одну акцію: 1000 грн <br> У власності: 0</p><input type="text"
id="stock6" placeholder="Кількість для покупки">
        <button
onclick="BuyStock(1,3,document.getElementById('stock6').value)">Покупка</button
>
        <button onclick="BuyStock(1,3,-
document.getElementById('stock6').value)">Продажа</button>
    </div>
</div>
<div id="highrisk">

```

```

<h1>Високий ризик</h1>
<div class="stock">
  <h3>Payoneer – 0</h3>
  <p>За одну акцію: 10 грн <br> У власності: 0</p><input type="text"
id="stock7" placeholder="Кількість для покупки">
  <button
onclick="BuyStock(2,1,document.getElementById('stock7').value)">Покупка</button
>
  <button onclick="BuyStock(2,1,-
document.getElementById('stock7').value)">Продажа</button>
</div>
<div class="stock">
  <h3>Amazon – 0</h3>
  <p>За одну акцію: 100 грн <br> У власності: 0</p><input type="text"
id="stock8" placeholder="Кількість для покупки">
  <button
onclick="BuyStock(2,2,document.getElementById('stock8').value)">Покупка</button
>
  <button onclick="BuyStock(2,2,-
document.getElementById('stock8').value)">Продажа</button>
</div>
<div class="stock">
  <h3>Apple – 0</h3>
  <p>За одну акцію: 1000 грн <br> У власності: 0</p><input type="text"
id="stock9" placeholder="Кількість для покупки">
  <button
onclick="BuyStock(2,3,document.getElementById('stock9').value)">Покупка</button
>
  <button onclick="BuyStock(2,3,-
document.getElementById('stock9').value)">Продажа</button>

```

```

    </div>
  </div>
</div>
</div>

```

Блок у файлі index.html який відповідає за отримання позики:

```

<h1 class="title">Позики</h1>
<div id="loanbuttons">
  <button onclick="setLoan(100000,.05,true,60000)">1,000.00 грн під
5%</button>
  <button onclick="setLoan(100000,.02,false,60000)">1,000.00 грн під
2%</button>
  <button onclick="setLoan(250000,.1,true,60000)">2,500.00 грн під 10%
</button>
  <button onclick="setLoan(500000,.03,false,60000)">5,000.00 грн під 3%
</button>
  <button onclick="setLoan(1000000,.1,true,60000)">10,000.00 грн під 10%
</button>
  <button onclick="setLoan(1000000,.05,false,60000)">10,000.00 грн під 5%
</button>
</div>
<div id="afterloan" style="display: none;">
  <input type="text" id="afterloan-payamm"></input>
  <button onclick='payLoan(document.getElementById("afterloan-
payamm").value)'>Розплатитися</button>
</div>

```

Блок у файлі index.html для оформлення авто платежів:

```

<div style="display: none" class="menu">
  <h1 class="title hasBusinessName">&zwnj;</h1>
  <button onclick="realiseProd(1)">Зробити <span
class="hasBusinessProduct">&zwnj;</span></button>
  <button onclick="autoBuyer()">Купити авто-покупку</button>
  <p style="display:inline" id="autoMakerCost">Ціна: 100 грн</p>
  <p id="unsoldProductInfo">Продавно: 0</p>
  <button onclick="addPrice(-1)">Нижче</button>
  <button onclick="addPrice(1)">Вище</button>
  <p id="pricePerProduct" style="display: inline;">Ціна за <span
class="hasBusinessProduct">&zwnj;</span>: грн0.05
</p>
  <p id="demandinfo">Попит: 10%</p>
  <button onclick="purchaseStocks()">Купити Маркетинг</button>
  <p id="marketingCost" style="display: inline;">Ціна: 100 грн</p>
  <br><br>
  <button onclick="purchaseMats(1)">Покупка</button>
  <p id="matsinfo" style="display: inline;">Матеріали: 0</p>
  <p id="matsprice">Ціна: 20 грн</p>
</div>

```

Функція оновлення акцій у файлі index.js:

```

const stocksUpdate = () => {
  const riskLevels = [
    { chance: LOWRISKCHANCE, change: LOWRISKMAXCHANGE },
    { chance: MEDRISKCHANCE, change: MEDRISKMAXCHANGE },
    { chance: HIGHRISKCHANCE, change: HIGHRISKMAXCHANGE },
  ];

```

```

for (let risk = 0; risk < 3; risk++) {
  const { chance, change } = riskLevels[risk];

  for (let i = 1; i < 4; i++) {
    const stockElement =
      stockscontainerHTML.children[risk].children[i].children[0];
    const priceElement =
      stockscontainerHTML.children[risk].children[i].children[1];
    const rand = Math.random().toFixed(2);

    if (rand == chance) {
      stockElement.className = "";
      stockElement.innerText = `${stocks[risk][i - 1].name} – 0`;
      continue;
    }

    const isDecrease = rand > chance;
    const vc = (isDecrease ? -1 : 1) * Math.random() * change;
    stockElement.className = isDecrease ? "stockdec" : "stockinc";
    stockElement.innerText = `${stocks[risk][i - 1].name} ${
      isDecrease ? "▽" : "△"
    } ${asDollars(Math.trunc(vc))}`;

    stocks[risk][i - 1].price = Math.trunc(
      Math.max(1, stocks[risk][i - 1].price + vc)
    );
    priceElement.innerHTML = `За акцію: грн${asDollars(
      stocks[risk][i - 1].price
    )}<br>У власності: ${stocks[risk][i - 1].owned}`;
  }
}

```

```

    }
};

```

Функція придбання акцій у файлі index.js:

```

const BuyStock = (risk, index, amount) => {
    const asInt = parseInt(amount);

    if (isNaN(asInt)) {
        logError(`${amount} не є інтеджером`);
        return;
    }

    const { name, price, owned } = stocks[risk][index - 1];

    if (funds < asInt * price) {
        logError(`купити не вдалося ${amount} ${name} бо коштів мало`);
        return;
    }

    if (asInt < 0 && -asInt > owned) {
        logError(`Ви не повинні ${amount} ${name}`);
        return;
    }

    addFunds(-asInt * price);
    stocks[risk][index - 1].owned += asInt;

    const priceElement =
        stockscontainerHTML.children[risk].children[index].children[1];

```

```

priceElement.innerHTML = `За акцію: грн${asDollars(
  price
)}<br>У власності: ${stocks[risk][index - 1].owned}`;
};

```

Функція оплати позики у файлі index.js:

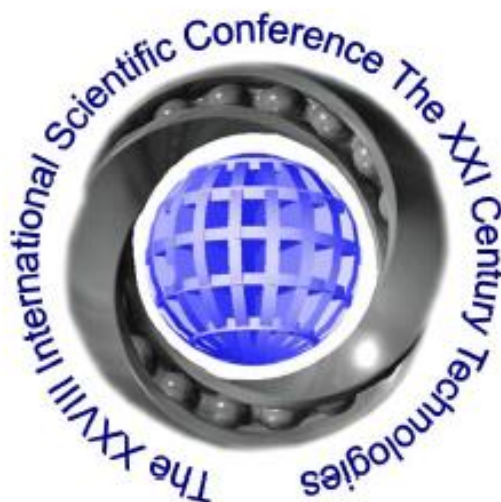
```

const payLoan = (quantity) => {
  const asInt = parseInt(quantity);
  if (isNaN(asInt)) {
    logError(`${quantity} не є числом`);
    return;
  }
  const payment = asInt * 100;
  if (funds >= payment) {
    addFunds(-payment);
    loan.amount -= payment;
    loanHTML.innerText = `Кредит: ${loan.amount} залишилось заплатити`;
    logSuccess(`Сплачено ${quantity}`);
  } else {
    logError(`у вас недостатньо ${quantity} тисяч`);
  }
  if (loan.amount <= 0) {
    addFunds(-loan.amount);
    initLoan();
  }
};

```

Тези

-  Сумський національний аграрний університет
-  Національний технічний університет «КПІ»
-  Політехніка Свентокржинська в Кельцах (Польща)
-  ТОВ «ТРІЗ»
-  Сумський державний університет
-  Державний біотехнологічний університет
-  Українська технологічна академія

**ТЕХНОЛОГІЇ ХХІ СТОРІЧЧЯ**

Збірник тез за матеріалами 28^{ої} міжнародної науково-
практичної конференції
(23-25 листопада 2022 р.)

Частина 2

Секції: «Прогресивні технології в економіці»,
«Прогресивні технології в екології та енергоефективність»,
«Прогресивні технології у педагогіці»

Суми – 2022

<i>Мартинченко С.В., студентка, Агаджанов-Гонсалес К.Х., ст.викладач, СНАУ, Суми, Україна</i>	
СУЧАСНІ НАПРЯМКИ ЕКСТРАКЦІЇ МЕТАДАНИХ ДЛЯ СТРУКТУРУВАННЯ ТА ПОШУКУ ПРЕЗЕНТАЦІЙ	166
<i>Ребрій А.М., Сумський національний аграрний університет, Україна</i>	
ОСОБЛИВОСТІ ДИСТАНЦІЙНОГО НАВЧАННЯ ПІДГОТОВКИ СТУДЕНТІВ В АГРАРНІЙ ОСВІТІ	167
<i>Ребрій А.М., Сумський національний аграрний університет, Україна</i>	
ТЕХНОЛОГІЇ МОДЕЛЮВАННЯ ТА ВІЗУАЛІЗАЦІЯ ГЕОМЕТРИЧНИХ ОБ'ЄМНИХ ТІЛ	168
<i>Рибенко І.О., Сумський національний аграрний університет, Україна</i>	
СУЧАСНІ ТЕНДЕНЦІЇ ВИКЛАДАННЯ ГРАФІЧНИХ ДИСЦИПЛІН	169
<i>Рибенко І.О., Сумський національний аграрний університет, Україна</i>	
КОМП'ЮТЕРНА ГРАФІКА - ІНСТРУМЕНТ ДЛЯ ПІДВИЩЕННЯ ЯКОСТІ ЗНАНЬ	170
<i>Захарова І.О., к.п.н., доцент, СДПУ імені А.С.Макаренка, м.Суми, Україна</i>	
ПЕДАГОГІЧНІ ОСОБЛИВОСТІ ДИСТАНЦІЙНОГО НАВЧАННЯ СТУДЕНТІВ У ЗАКЛАДІ ВИЩОЇ ОСВІТИ	171
<i>Семерня О.В., Сумський національний аграрний університет</i>	
РОЛЬ ВИКЛАДАЧА В ОРГАНІЗАЦІЇ САМОСТІЙНОЇ ПІДГОТОВКИ ЗДОБУВАЧІВ ВИЩОЇ ОСВІТИ З ПИТАНЬ ОХОРОНИ ПРАЦІ	173
<i>Єременко О. В., студент, Агаджанова С.В., к.т.н., доцент, СНАУ, Суми, Україна</i>	
ВИКОРИСТАННЯ ІНТЕРАКТИВНИХ ВЕБ-ТРЕНАЖЕРІВ В НАВЧАЛЬНОМУ ПРОЦЕСІ	176
<i>Гончар Д., Лапченко І., студенти, СНАУ, Суми, Україна</i>	
ІНФОРМАЦІЙНІ СИСТЕМИ І ТЕХНОЛОГІЇ У ОСВІТІ	177
<i>Статівка В.І., д.п.н., професор, Захарова І.О., к.п.н., доцент, СумДПУ ім. А.С. Макаренка, м. Суми, Україна</i>	
УПРАВЛІННЯ НАУКОВИМ ДОСЛІДЖЕННЯМ СТУДЕНТА: СПОСОБИ УПЕРЕДЖЕННЯ ПОМИЛОК У СТВОРЕННІ ТЕКСТОВОГО ВАРІАНТУ	178
<i>Баталова А.Б., старший викладач, СНАУ, Суми, Україна</i>	
МУЛЬТИМЕДІА, ЯК ЗАСІБ НАВЧАННЯ СТУДЕНТІВ В УМОВАХ ВІЙСЬКОВОГО СТАНУ.....	180
<i>Хурсенко С.М., к.ф.-м.н., доцент, СНАУ</i>	
САМОРОЗВИТОК ОСОБИСТОСТІ В УМОВАХ КОМПЕТЕНТНІСНО-ОРІЄНТОВАНОГО ПІДХОДУ	181
<i>Хурсенко С.М., к.ф.-м.н., доцент, СНАУ</i>	
МУЛЬТИМЕДІЙНІ ТЕХНОЛОГІЇ – ЗАСІБ НАВЧАННЯ НОВОГО ПОКОЛІННЯ	183
<i>Хіманова А.О., СНАУ, Україна</i>	
ВИКОРИСТАННЯ СИМУЛЯТОРІВ У ОСВІТНІЙ ДІЯЛЬНОСТІ УНІВЕРСИТЕТУ	184

УДК 004.42

Хіманова А.О., СНАУ, Україна

ВИКОРИСТАННЯ СИМУЛЯТОРІВ У ОСВІТНІЙ ДІЯЛЬНОСТІ УНІВЕРСИТЕТУ

Вступ. В останні роки використанню ігрових технологій із застосуванням ЕОМ в освіт-ньому процесі педагогічних закладів вищої освіти приділено значну увагу. Ігрові технології відкрили нові можливості до шляху скорочення часу занять, розширення засобів самостійно-го навчання, підвищення якості викладання тощо. Ігрові технології поряд з іншими метода-ми навчання служать накопиченню професійного досвіду, близько до реального, і, по суті, замінюють досвід лабораторних та практичних робіт. Крім того, цей метод подекуди діє краще, ніж інші методи пізнання.

Виклад основного матеріалу дослідження. Розвиток творчого потенціалу, здатності до прийняття рішень та інтелектуальних здібностей потребує перегляду мети та організації освітнього процесу: він має бути більш динамічним, гнучким та пов'язаним з ініціативною поведінкою суб'єктів навчання.

В реальному часі майже всі симулятори комп'ютеризовані, та включають в себе багато-рівневі алгоритми, які оцінюють ефективність дій на основі прийнятих рішень. Більшість симуляторів базуються на реальних процесах і як результат використовують реальні дані, щоб бути якомога точнішими та давати реалістичний досвід.

Використання моделей у навчальному процесі не вважається новим методом. Моделі ви-користовуються здавна. Симулятори використовуються у всіх галузях навчального процесу, починаючи з початкових класів до вищих навчальних закладів. Ще однією перевагою симуляторів є те, що вони є дешевою альтернативою реальних об'єктів. Симулятори дають можливість без будь-яких інструментів моделювати фізичні процеси та віртуально проводити віртуальні лабораторні заняття. Це само собою не лише економить фінанси, а й дає можливість обійтися без них. Перевага симуляторів, це те, що вони безпечні у використанні. Проведення деяких досліджень загрожує людині летальними наслідками.

За допомогою симуляторів студенти, хоч і віртуально, мають можливість втілити у жит-тя вивчені під час занять знання. У ході проведення цих досліджень студенти зміцнюють те оретичні знання та вносять свою користь у розвиток впровадження їх в життя. Крім того ро-блять свій внесок у розвиток та прогресування самих симуляторів, підвищення рівня симуля-торів до того рівня, коли вони даватимуть результати однакові з реальними дослідженнями. З часом студенти вже є слухачами, а стають учасниками науково-дослідницьких робіт. А це, у свою чергу, збільшує інтерес студентів вчитися та проводити наукові дослідження.

Перевагою симуляційних технологій є те, що їх впровадження дозволяє піти від традиційних форм освітнього процесу на семінарах, практичних заняттях, де в центрі уваги перебуває викладач, дозволяє змістити акцент на студента, надавши останньому можливість відпрацьовувати навички, припускатися помилок та виправляти їх, аналізувати ситуацію і робити висновки. Застосування методів активного навчання, зокрема, симуляційної рольової гри, дозволяє студентам залишаючись в позиції учнів, виконувати професійні дії і вчинки. Форма організації такої діяльності практично відтворює форми реальної професійної діяльності. Тому для викладача будь-яка технологія має практичний сенс тією мірою, в якій допомагає йому здійснювати цілеспрямований вибір відповідного методу навчання або їх поєднання для вирішення конкретних дидактичних завдань.

Висновки. Таким чином, ігрова діяльність виконує такі функції: спонукальну (викликає інтерес у студентів); комунікабельну (засвоєння елементів культури спілкування майбутніх спеціалістів); самореалізації (кожен учасник гри реалізує свої можливості); розвивальну (розвиток уваги, волі та інших психічних якостей); розважальну (отримання задоволення); діагностичну (виявлення відхилень у знаннях, уміннях та навичках, поведінці); корекційну (внесення позитивних змін у структуру особистості майбутніх фахівців). Ігрові методи багатопланові, і кожен з них у той чи інший спосіб сприяє виробленню певної навички. З огляду на це, виокремлюють ігри-вправи, ігрові дискусії, ігрові ситуації, рольові та ділові навчальні ігри, комп'ютерні ділові ігри.

Навчальна симуляційна гра, у нашому розумінні, найбільш повно відповідає уявленню про новий, контекстний тип навчання, оскільки в ній знаходять своє відображення найбільш характерні його риси. У зв'язку з цим основне завдання навчальної симуляційної гри не тільки імітувати реальні умови професійної діяльності в навчанні, але й забезпечити можливості для відтворення або імітації тих протиріч, з якими стикається фахівець на шляху пізнання та професійної діяльності.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

МАТЕРІАЛИ

науково-практичної конференції
викладачів, аспірантів та студентів
Сумського НАУ

(25-28 квітня 2023 р.)

Рудакова Д.О. РОЛЬ ЕКОЛОГІЧНОГО МАРКЕТИНГУ В ФОРМУВАННІ КОНКУРЕНТНИХ ПЕРЕВАГ ДЕРЖАВНОГО ПІДПРИЄМСТВА	339
Базарний С.О. УПРАВЛІННЯ ТОВАРНОЮ ПОЛІТИКОЮ НА ПІДПРИЄМСТВАХ МАЛОГО БІЗНЕСУ	340
Сергієнко В.Г. КАДРОВА БЕЗПЕКА ЯК ОСНОВНИЙ ЕЛЕМЕНТ ЕКОНОМІЧНОЇ БЕЗПЕКИ ПІДПРИЄМСТВА	341
Бельська А.О. РЕКЛАМНА ДІЯЛЬНІСТЬ НА ПІДПРИЄМСТВАХ РОЗДРІБНОЇ ТОРГІВЛІ ТА СПОСОБИ ЇЇ ВДОСКОНАЛЕННЯ	342
Дундук Д.С. РОЛЬ І ЗНАЧЕННЯ КРЕДИТУ В РОЗВИТКУ ЕКОНОМІКИ УКРАЇНИ	343
В'юненко О.Б. СУЧАСНІ ПРОБЛЕМИ ІНФОРМАЦІЙНОЇ КІБЕРПІЄНИ	344
Баталова А.Б. ГЕОІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ЯК ІНСТРУМЕНТ ЕКОЛОГІЧНИХ ДОСЛІДЖЕНЬ	345
Маяк А.О. ЦИФРОВІ ІНСТРУМЕНТИ ДЛЯ ПУБЛІКАЦІЇ ТА ПРОДАЖУ МУЗИКИ	346
Бусенко І.В. РОЗРОБКА ВІДЕОІГОР НА БАЗІ ТЕХНОЛОГІЇ P2E	347
Крамський А.С. ChatGPT – ТВІЙ МЕНТОР У НАВЧАННІ	348
Гервольський М.Д. АНАЛІТИЧНИЙ АНАЛІЗ РІЗНИХ ВИДИ ХОСТИНГІВ	349
Потапенко М.Т. АВТОМАТИЗОВАНА ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ СПОРТИВНОГО КОМПЛЕКСУ	350
Єременко О.В. РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ПЕРЕВІРКИ ПРАКТИЧНИХ РОБІТ ІЗ ВИБІРКИ ДАНИХ: ШЛЯХИ І ТЕХНОЛОГІЇ	351
Мотлюк М.В. А/В ТЕСТУВАННЯ ЯК МЕТОД МАРКЕТИНГОВИХ ДОСЛІДЖЕНЬ В ІТ	352
Олефіренко Д.В. ПРОГНОЗУВАННЯ КІЛЬКОСТІ ВЕЛИКОЇ РОГАТОЇ ХУДОБИ В УКРАЇНІ З ВИКОРИСТАННЯМ ЕКОНОМЕТРИЧНИХ МЕТОДІВ	353
Бабуров Д.В. ПРОГНОЗУВАННЯ КІЛЬКОСТІ СІЛЬСЬКОГОСПОДАРСЬКОЇ ПТИЦІ В УКРАЇНІ З ВИКОРИСТАННЯМ ЕКОНОМЕТРИЧНИХ МЕТОДІВ	354
Шакотько А.М. ІНФОРМАЦІЙНА ПІДСИСТЕМА МАТЕРІАЛЬНОГО ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ВСП «ГАФК СНАУ»	355
Басанець Д.В. ІНФОРМАЦІЙНА ПІДСИСТЕМА ОБЛІКУ КАДРОВОГО ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ КОЛЕДЖУ ВСП «ГАФК СНАУ»	356
Хіманова А.О. ПРОБЛЕМИ УПРАВЛІННЯ ІТ ПРОЄКТАМИ СУЧАСНИХ НАВЧАЛЬНИХ СИСТЕМ	357
Сіденко А.С. ПРОБЛЕМИ ПОБУДОВИ САЙТІВ ДЛЯ СИСТЕМ ДИСТАНЦІЙНОГО НАВЧАННЯ ...	358
Шакотько Є.С. СУЧАСНІ АЛГОРИТМИ АНАЛІЗУ ПРОДУКТИВНОСТІ ТА МОНИТОРИНГУ КОМП'ЮТЕРНИХ МЕРЕЖ	359
Грошовик Р.Ю. ОБГРУНТУВАННЯ НЕОБХІДНОСТІ АВТОМАТИЗАЦІЇ ВЕДЕННЯ ПОГОСПОДАРСЬКОГО ОБЛІКУ ТЕРИТОРІАЛЬНИХ ГРОМАД	360
Наконечний О.Б. АНАЛІЗ ПРОБЛЕМИ АВТОМАТИЗАЦІЇ РОЗРАХУНКУ БАЛАНСУ ГУМУСУ І ПОЖИВНИХ РЕЧОВИН В ҐРУНТІ ТА ПРОПОЗИЦІЇ ЩОДО ЇЇ ВИРІШЕННЯ	361
Мартинченко С.В. ОБРОБКА ВЕЛИКИХ ДАНИХ: ПРОБЛЕМИ І ШЛЯХИ ЇХ ВИРІШЕННЯ	362
Zhang Yuanyi THE KEY FACTORS IN PERSONAL DEVELOPMENT FROM THE PERSPECTIVE OF ENTERPRISES	363
Qin Fan IMPROVEMENT OF THE RISK MANAGEMENT SYSTEM AT THE ENTERPRISE	364
Jiang Maoli ORGANIZE THE CONCEPT AND METHOD OF BUSINESS PROCESS IMPROVEMENT	365
Zhang Buyao DISCUSSION ON THE MANAGEMENT OF CAREER PROMOTION OF ENTERPRISE EMPLOYEES	366
Cui Xiaoping THE IMPACT OF INNOVATIVE TRANSFORMATIONS IN THE AGRO-INDUSTRIAL SPHERE ON THE ACHIEVEMENT OF THE SUSTAINABLE DEVELOPMENT GOALS	367
Zhao Jing CONCEPTUAL APPROACHES TO MANAGING THE BEHAVIOR OF THE COMPANY'S PERSONNEL ON THE BASIS OF CORPORATE CULTURE	368
Гулько Б. В. МЕХАНІЗМ ДЕРЖАВНОГО РЕГУЛЮВАННЯ В СФЕРІ СТАЛОГО РОЗВИТКУ	369
Маслак О.М., Санін Р.С. ІНСТРУМЕНТИ ДЕРЖАВНОГО РЕГУЛЮВАННЯ ФЕРМЕРСЬКИХ ГОСПОДАРСТВ УКРАЇНИ В УМОВАХ ВОЄННОГО СТАНУ	370
Ніколаєнко Т.С. НАПРЯМИ РОЗВИТКУ ПУБЛІЧНОГО АДМІНІСТРУВАННЯ ОСВІТИ НА РІВНІ ТЕРИТОРІАЛЬНОЇ ГРОМАДИ	371
Гребченко Т.А., Стеценко Б.М. КОМУНІКАЦІЙНИЙ ЗВ'ЯЗОК ОРГАНІВ ДЕРЖАВНОГО УПРАВЛІННЯ ІЗ СУСПІЛЬСТВОМ	372
Майданевич С.В. АДМІНІСТРАТИВНО-ПРОЦЕДУРНІ ЗАСАДИ РЕГУЛЮВАННЯ В БАНКІВСЬКІЙ СФЕРІ	373
Chonglong Bi MANAGEMENT OF ORGANIZATIONAL CULTURE AS A FACTOR OF INCREASING COMPETITIVENESS	374
Тимошенко М.І. ПРИНЦИПИ СТВОРЕННЯ ТА УПРАВЛІННЯ КЛАСТЕРНИМИ ТА НАДКЛАСТЕРНИМИ ЛІКАРНЯМИ	375

ПРОБЛЕМИ УПРАВЛІННЯ ІТ ПРОЄКТАМИ СУЧАСНИХ НАВЧАЛЬНИХ СИСТЕМ

Хіманова А.О., студ. 4 (2ст) курсу ФЕіМ, спец. «Інформаційні системи та технології»

Науковий керівник: к.е.н., доц. В'юненко О.Б.

Сумський НАУ

Управління проектом створення навчальної системи симулятора може бути складною задачею, яка потребує вирішення багатьох проблем. Деякі з них включають недостатню комунікацію між членами команди, недостатню оцінку ризиків, недостатнє планування, недостатнє управління змінами, проблеми з управлінням бюджетом та проблеми з відповідальністю. Щоб успішно керувати проектом, необхідно забезпечити ефективну комунікацію, враховувати можливі ризики, мати чіткий план та графік робіт, встановити процес управління змінами, відстежувати витрати та визначати точну роль та відповідальність кожного члена команди. Управління проектом може бути викликом, але за допомогою правильного підходу може бути успішно реалізованою задачею.

Для досягнення стабільних позитивних результатів проєкті повинні бути легкокерованими. Розробка легкокерованих проєктів вимагає діяльності в двох паралельних напрямках. По-перше, необхідно залучити всіх, хто робить ставку на проєкт, до визначенні конкретних цілей проєкту і засобів їх досягнення. По-друге, необхідний пошук такого варіанту, який би забезпечував економне витрачання ресурсів при реалізації проєкту.

Управління проєктами - складний процес, особливо коли розробляється навчальна система симулятора. Ця стаття зосереджується на проблемах, які можуть виникнути під час управління проектом створення навчальної системи симулятора та пропонує рішення для їх вирішення.

Першою проблемою є недостатня комунікація між членами команди. Недостатня комунікація може призвести до неправильного розуміння вимог та несвоєчасного виявлення проблем. Рішенням може бути створення каналів комунікації та регулярні зустрічі, щоб встановити ясні очікування та взаємодію між членами команди.

Другою проблемою є недостатня оцінка ризиків. Управління ризиками - це важлива складова частина проєктного менеджменту, оскільки недооцінка ризиків може привести до затримок, додаткових витрат та втрати якості. Рішенням може бути проведення відповідної оцінки ризиків та розроблення плану управління ризиками, щоб зменшити їх вплив.

Третьою проблемою є недостатнє планування. Недостатнє планування може призвести до неправильного розуміння вимог та невиконання планів. Рішенням може бути створення чіткого плану та графіка робіт, які відображатимуть реалістичний хронометраж робіт та дозволять вчасно виявляти проблеми.

Четвертою проблемою є недостатнє управління змінами. Під час проекту можуть виникати зміни, що можуть впливати на виконання проекту. Недостатнє управління змінами може призвести до затримок та додаткових витрат. Рішенням може бути створення процедур управління змінами, які будуть включати оцінку впливу змін на проект, затвердження змін та визначення плану впровадження змін.

Останньою проблемою є недостатнє управління ресурсами. Недостатнє управління ресурсами може призвести до неефективного використання ресурсів та затримок у проекті. Рішенням може бути створення детального плану використання ресурсів, що включатиме у себе графік робіт та розподіл ресурсів на різні етапи проекту.

Управління проектом створення навчальної системи симулятора може бути складним процесом, але врахування та вирішення вищезазначених проблем може допомогти уникнути затримок, додаткових витрат та втрати якості. Для успішного виконання проекту важливо мати ефективну комунікацію між членами команди, оцінювати ризики, планувати роботу, управляти змінами та ефективно використовувати ресурси.

Другим важливим аспектом управління проектом створення навчальної системи симулятора є визначення якості. Якість повинна бути визначена та включена у план робіт з самого початку проекту, щоб запобігти можливим проблемам на пізніших етапах. У процесі виконання проекту повинна проводитись систематична перевірка якості продукту та процесу, що дозволить вчасно виявити проблеми та виправити їх.

Нарешті, важливо забезпечити належне документування проекту. Документація дозволить зберегти знання та досвід, набуті під час проектування та розробки системи, а також забезпечити передачу цих знань іншим членам команди в майбутньому. Документування повинно бути проведено на кожному етапі проекту та містити інформацію про процес, прийняті рішення, результати тестування та оцінку якості.

Отже, успішне управління проектом створення навчальної системи симулятора включає в себе ефективне управління комунікаціями, ризиками, роботою, змінами та ресурсами, а також визначення та відстеження якості продукту та процесу, увагу до потреб користувачів та належне документування проекту.