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

БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА

на тему:

Комп'ютерна підсистема для визначення термінів виконання етапів бакалаврських робіт на основі діаграми Ганта

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав студент 4 курсу, групи КІ-20мс
спеціальності123 — Комп'ютерна інженерія
Білик Т. В.

Керівник к.т.н., доц. каф. ОТ
Снігур А. В.
"15" 06 2022 р.

Рецензент к.т.н., доц. каф. ЗІ
Куперштейн Л. М.
"16" 06 2022 р.

Допущено до захисту
д.т.н., проф. Азаров О.Д.
"17" 06 2022 р.

ВНТУ 2022

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

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

ЗАТВЕРДЖУЮ
Завідувач кафедри О.Д. Азаров
"09" лютого 2022року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ

студенткці Білик Тетяні Василівні

1 Тема роботи «Комп'ютерна підсистема для визначення термінів виконання етапів бакалаврських робіт на основі діаграми Ганта.» керівник роботи Снігур А.В к.т.н., доцент, затверджено наказом вищого навчального закладу від "24" березня 2022року №66

2 Строк подання студентом роботи 17.06.2022

3 Вихідні дані до роботи :засоби – інтегроване середовище розробки java. Інтерфейс програми – додаток для платформи java, який розроблений на мові програмування Java. Передбачено створення бази даних для збереження даних користувачів.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналітичний огляд існуючих підходів та програмних засобів які використовують діаграму Ґанта для планування. Особливості створення програмного забезпечення для планування захистів дипломних робіт на основі діаграми Ґанта. Програмне забезпечення для врахування змін у термінах готовності дипломних робіт на основі діаграми Ґанта. Висновки. Література. Додатки.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): лістинг програми.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1— Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Снігур А.В. к. т. н., доцент каф. ОТ		

7 Дата видачі завдання _____

8 Календарний план виконання БДР приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Підпис
1	Постановка задачі роботи	09.03.2022	вик.
2	Огляд та аналіз сучасних програмних забезпечень для планування за розкладом на основі діаграми Ґанта	10.03.2022 – 16.03.2022	вик.
3	Аналіз ринку java-додатків планування на основі діаграми Ґанта	17.03.2022 – 18.03.2022	вик.
4	Розробка структури бази даних, маршрутизації додатку. Розробка реєстрації та авторизації користувачів. Розробка інтерфейсу java –додатків	19.03.2022 – 18.04.2022	вик.
5	Програмна реалізація структури програмного забезпечення для планування на основі діаграми Ґанта	19.04.2022– 08.05.2022	вик.
6	Підготовка матеріалів для опису програмного забезпечення для планування на основі діаграми Ґанта	09.05.2022 – 22.05.2022	вик.
7	Оформлення пояснювальної записки	23.05.2022 – 01.06.2022	вик.
8	Перевірка якості виконання бакалаврської роботи	02.06.2022 – 08.06.2022	вик.

Студент _____ Білик Тетяна Василівна
(підпис)

Керівник роботи _____ Снігур Анатолій Васильович
(підпис)

АНОТАЦІЯ

Дипломна робота викладена на 66 сторінках, вона містить 14 ілюстрацій, 3 таблиць, 14 джерел в переліку посилань.

Об'єктом розгляду є мктодичне та лабораторне забезпечення ряду дисциплін, що викладаються в ВНТУ.

Предмет роботи — створення програмного забезпечення для визначення термінів виконання етапів бакалаврських робіт на основі діаграми Ганта.

Метою роботи є впровадження розробленого програмного додатку для ефективного планування часу готовності до захистів дипломних робіт.

У першому розділі розглядаються існуючі підходи та програмні засоби, у яких використовують діаграму Ганто для планування. У другому розділі - аналізуються особливості створення програмного забезпечення для планування за розкладом на основі діаграми Ґанта. В розділі третьому розглядається інтегроване середовище розробки Intellij IDEA у якому розроблена комп'ютерна підсистема для визначення термінів виконання етапів бакалаврських робіт на основі діаграми Ганта. Розроблено програмне забезпечення яке надає змогу перевірити готовність студентів до захистів бакалаврських робіт на основі діаграм Ганто і проводиться тестування роботи інтерфейсу програми.

За резутьтатами роботи зроблено висновки та представлено інструкцію по користувачеві.

Ключові слова: діаграма, Ганто, додаток, Java, дипломна, студенти, захист, готовність.

ABSTRACT

Thesis is presented on 66 pages, it contains 14 illustrations, 3 tables, 14 sources in the list of references.

The object of consideration is the methodological and laboratory support of a number of disciplines taught at VNTU.

The subject of the work is the creation of software for determining the timing of the stages of bachelor's theses on the basis of the Gantt chart.

The aim of the work is to implement the developed software application for effective planning of the time of readiness for the defense of theses.

The first section discusses existing approaches and software tools that use the Gantt chart for planning. The second section analyzes the features of creating software for scheduling on the basis of the Gantt chart. The third section discusses the integrated development environment Intellij IDEA, which has developed a computer subsystem to determine the timing of the stages of undergraduate work based on the Gantt chart. Software has been developed to test students' readiness for bachelor's theses based on Gantt charts, and testing of the program's interface is being conducted.

Based on the results of the work, conclusions were made and instructions for the user were presented.

Keywords: diagram, Gantt, application, Java, diploma, students, defense, readiness.

ЗМІСТ

ВСТУП

В даний час технічний прогрес все частіше і частіше зустрічається у повсякденному житті людини, в побуті, в медицині, у виробництві, у сфері обслуговування та освіти. Одночасно розвиваються й Інтернет технології. За будьякою інформацією найчастіше люди звертаються саме до Інтернету або до електронних книг.

Мова Java з'явилася як частина проекту створення передового програмного забезпечення (ПО) для різних побутових приладів. Реалізація програми була почата на мові С ++, але з часом виник ряд проблем, найкращим засобом боротьби з якими була зміна самого інструмента - мови програмування. Стало очевидним, що потрібно платформонезалежну мову програмування, яка дає можливіть створювати програми, які не доводилося б компілювати окремо для кожної архітектури і можна було б використовувати на різних процесорах під різними операційними системами.

У розробленому додатку є функція, яка після завершення вивчення будь-якого предмету, надає змогу зобразити діаграму Ґанта, яка показує запланований та реальний час готовність студентів до захисту дипломного проекту.

Актуальність теми полягає в тому, що зі стрімким розвитком технологій та появою великого обсягу статистичних даних з'явились складнощі з їх опрацюванням, складанням загальної статистики та її відображенням у вигляді, який має інтуїтивно зрозумілий інтерфейс для користувача, який оперує цими даними. Кожний крок вимагає від користувача постійної концентрації та уваги, дані необхідно вводити вручну у поля, які розташовані на різних частинах документу.

Тому метою є розробка програмного додатку, який має необхідні функції для додавання та відображення даних, та в свою чергу зможе полегшити обробку, автоматично уструктуризувати отриману від користувача інформацію. Програма розроблася за допомогою мови програмування Java.

Для досягнення поставленої мети потрібно здійснити наступні задачі:

— провести аналіз методів та технологій розробки;

— провести аналіз засобів статичного та динамічного планування;

— проаналізувати способи організації діаграми Ґанта;

— обрати інтегроване середовище розробки додатку;

— розробити алгоритм функціонування додатку;

— проаналізувати існуючі аналоги, а також особливості реалізації додатків;

— програмно реалізувати та протестувати розроблений додаток.

Об’єкт дослідження — Комп'ютерна підсистема для визначення термінів виконання етапів бакалаврських робіт на основі діаграми Ганта.

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

Практичне значення отриманих результатів полягає в можливості ефективного безперервного планування та прийняття рішень у режимі реального часу.

1 АНАЛІТИЧНИЙ ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ ТА ПРОГРАМНИХ ЗАСОБІВ ЯКІ ВИКОРИСТОВУЮТЬ ДІАГРАМУ ҐАНТА ДЛЯ ПЛАНУВАННЯ

1.1 Моделі визначення пріоритетів при виконанні робіт

При вирішенні певних завдань або задоволенні потреб користувачів виникають питання про порядок їх виконання: які питання взагалі потрібно планувати, а які можна відкласти надовго. Оскільки неможливо здійснити всі очікування та необхідні зміни відразу, необхідно спочатку визначити, які завдання необхідно вирішувати, щоб задовольнити найважливіші та нагальні потреби чи вирішити найбільш серйозні проблеми.

Важливість завдання залежить від того, наскільки його виконання сприяє досягненню мети. Якщо завдання є обов'язковим і мета не може бути досягнута без вирішення проблеми, визначте рівень важливості як «високий». Рівень важливості визначається як «помірний», якщо проблему можна частково вирішити без будь-яких дій. Щоб визначити важливість, потрібно поставити запитання: «Якщо завдання не виконано, то що найгірше?». Відповідь «не неправильно» визначає низьку важливість.

Терміновість визначає необхідність вирішити проблему або вжити заходів протягом певного періоду часу. Теоретично можна сказати, що чим менше часу витрачається на певні рішення та дії, тим вище кореляція [2].

При виконанні роботи аналізуються два різних типи пріоритетів: статичні та динамічні. Статичні плани використовуються для розв'язування задач з повною інформацією: часом виконання, структурою і типом часу, характеристиками їх взаємодії та способами обміну інформацією. У цьому випадку на етапі розробки системи ви можете створити статичний графік з інформацією про повний цикл виконання, а планування циклу виконання стає простим інтерпретатором для статичного графа.

Для простих алгоритмів системного планування, де ефективність не є критичним фактором, система пріоритетів не використовується. Однак у більшості

випадків ви використовуєте алгоритм планування на основі пріоритету завдання. Усі готові завдання мають пріоритет, а керівництво завжди має найвищий пріоритет. При визначенні пріоритетів завдань враховується їх часовий характер. Пріоритет може бути статичним або динамічним. Отже, за способом визначення пріоритету завдань алгоритми планування поділяються на алгоритми планування зі статичним пріоритетом і алгоритми планування динамічного пріоритету.

Завдання мають пріоритет, оскільки вони надходять в систему під час розробки і залишаються незмінними протягом усього життя системи.

Алгоритми, що використовують динамічний пріоритет, більш ефективні, але їх важко реалізувати. Алгоритми статичного пріоритету менш ефективні, але легші у реалізації.

У жорсткій системі реального часу час виконання кожного критичного завдання має бути гарантовано для всіх можливих сценаріїв застосування. Ви можете надати наступні гарантії:

— в результаті широкого тестування всіх можливих сценаріїв поведінки лідера об'єктні та керуючі програми;

— в результаті побудови статичного графіка;

— в результаті вибору математично обґрунтованого алгоритму динамічного планування.

При виборі алгоритму планування слід враховувати можливі залежності завдань. Завдання залежать, коли виникає один із двох типів залежностей: критичні розділи або обмеження виконання.

Проблема залежних від планування завдань дуже складна. Цю проблему можна вирішити наступними заходами:

— розділіть задачу планування на дві частини так, щоб існувала одна частина Конан заздалегідь;

— введення обмежувальних припущень щодо поведінки низки завдань.

При такому підході планування майже статичне.

Якщо всі терміни дотримані, розклад вважається правильним. Набір завдань [T_j] називається алгоритмом планування за алгоритмом планування, якщо цей алгоритм планування завжди забезпечує правильний план для цього набору завдань.

Розглянемо систему незалежних періодичних задач [T_j]. Також припустимо, що в цьому наборі завдань можуть бути деякі особливі випадки, наприклад Б. аперіодичні та спорадичні завдання.

Розкладом періодичних завдань буде таблиця із зазначенням того, яке завдання поставити в який час. Час створення цього розкладу відповідає гіперперіоду, який є найменшим спільним кратним усіх завдань.

Тому планувальник періодично повторює зазначені в розкладі послідовності виконання завдань.

Розглянемо приклад статичного розкладу, що містить серію з чотирьох задач: T_1 [*, 4, *, 1]; T_2 [*, 5, *, 1,8]; T_3 [*, 20, *, 1]; T_4 [*, 20, *, 2].

Гіперперіод цієї системи завдань становить 20. За цей час першим завданням потрібно керувати 5 разів, другим — 4, а третім і четвертим — по одному. При цьому всі терміни мають бути дотримані. Графік роботи такої системи завдань наведено в таблиці 1.

У таблиці 1.1. «I» означає завдання простою (idle task).

Планування такого роду зазвичай працює з перериваннями від таймера, завдання — це лише підпрограми, викликані обробником переривань у потрібний момент.

Оскільки таймери зазвичай запрограмовані не на генерацію переривання через певний час від початку зворотного відліку, а на переривання через певний час з певного моменту часу, зручніше представити таблицю в іншому вигляді: замість проміжків часу від початку циклу поміщати в неї відносні проміжки часу, від одного переривання до іншого.

Детальний приклад наведених завдань наведено в таблиці 1.1.

Модифікований приклад розкладу для наведеної системи задач наведено у таблиці 1.2.

Таблиця 1.1 — Приклад розкладу для наведеної системи задач

№	Час	Задача	№	Час	Задача
1	0	T_1	10	10.8	I
2	1	T_3	11	12	T_2
3	2	T_2	12	13.8	T_1
4	3.8	I	13	14.8	I
5	4	T_1	14	16	T_1
6	5	I	15	17	I
7	6	T_4	16	18	T_2
8	8	T_2	17	19.8	I
9	9.8	T_1			

Таблиця 1.2 — Модифікований приклад розкладу для наведеної системи задач

№	Час	Задача	№	Час	Задача
1	0.2	T_1	10	1	I
2	1	T_3	11	1.2	T_2
3	1	T_2	12	1.8	T_1
4	1.8	I	13	1	I
5	0.2	T_1	14	1.2	T_1
6	1	I	15	1	I
7	1	T_4	16	1	T_2
8	2	T_2	17	1.8	I
9	1.8	T_1			

Переваги такого типу алгоритму:

— надзвичайно проста передача управління завданнями;

— результати випробувань та перевірок дуже достовірні.

Через ці властивості цей тип алгоритму частіше використовується там, де потрібна висока надійність

Недоліки цього класу алгоритмів:

— відправлення фактично «від'єднано» від зовнішнього світу, оскільки обробляє переривання від таймера;

— розмір таблиці з розкладом може бути дуже великим при відповідному співвідношенні між періодами виконання завдань.

Будь-яка зміна (кількість завдань, час виконання тощо) вимагає перерахунку розкладу, а впоратися з спорадичними завданнями дуже важко.

Динамічне планування з динамічними пріорітетами. Є два типи планів з динамічним пріоритетом:

— EDF (earliest deadline first);

— LLF (least laxity first).

У плануванні EDF завдання визначають пріоритети за принципом, що «у будь-який момент часу найвищий пріоритет призначається завданню з найменшою відстанню від дати виконання». Модифікація відбувається з перенесенням завдання та без нього. У плані LLF пріоритет завдань призначається за принципом «у кожен момент завдання з найвищим пріоритетом займає найменше часу».

Доказ базується на наступному: якщо ви можете скласти правильний розклад для набору завдань, цей розклад завжди можна скоротити (і залишиться правильним), тобто порядок завдань буде таким же, як і при використанні EDF [3].

Ця ж теорема також стосується алгоритму LLF. Пояснимо тут поняття «резерв часу».

Резервний час — це різниця між часом, що залишився до кінцевого терміну, і часом, коли завдання ще потребує роботи.

L(t)=(d-t)-e (rem)

На цій діаграмі показано роботу завдання, готове в момент 2, з кінцевим терміном у момент 12. Завдання вирішується шляхом переміщення. Поки вона виконує свою роботу, часовий проміжок залишається тим самим, а кількість часу, який ще потрібно відпрацювати, такий ж, як термін що скорочується. Якщо завдання стає бездіяльним через заміну іншими завданнями вищого пріоритету, дедлайн

наближається, а час, який ще потрібно вирішити, залишається незмінним, тому в такі проміжки часового запасу часу зменшується.

На рисунку 1. 1 зображено діаграму, що ілюструє це поняття.

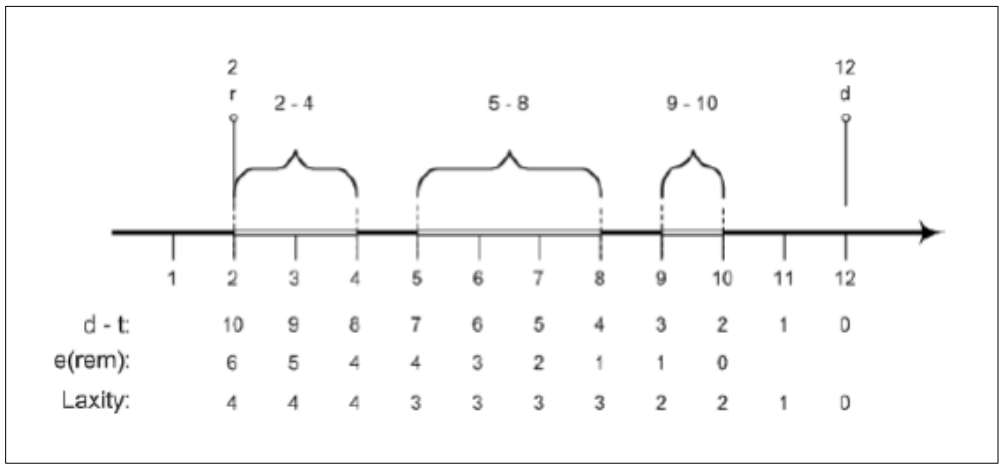


Рисунок 1.1 — Діаграма резерву часу

Алгоритм EDF без витіснення не є неоптимальним. Цю теорему можна довести, навівши зустрічний приклад, тобто поставивши набір задач, для яких, можна скласти правильний розклад, тоді як розклад, заданий алгоритмом планування EDF без зміщення, є неправильним.

Наприклад, маємо три задачі з параметрами:

На рисунку 1.2 зображено, який можна скласти розклад для даного набору.

На рисунку 1.3 зображено розклад, якщо використовувати алгоритм EDF без витіснення.

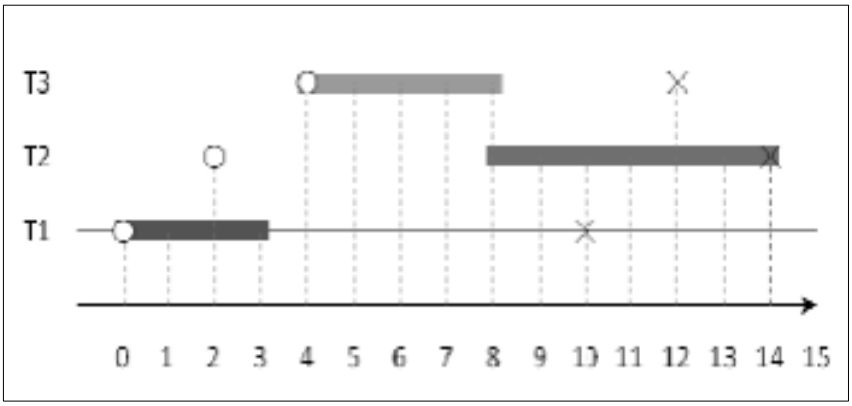


Рисунок 1.2 — Можливий розклад для вищенаведеної системи із трьох задач

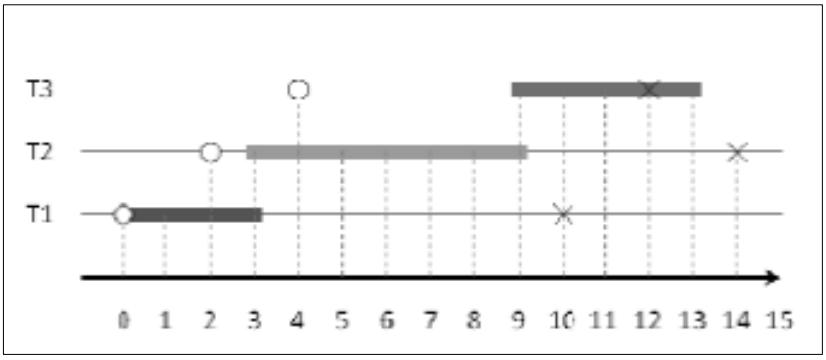


Рисунок 1.3 — Можливий розклад складений EDF-плануванням без витіснення

У момент 3 готове лише проблемне завдання 2, тому воно вибрано для виконання і не буде замінено до повного завершення його роботи, незважаючи на те, що в момент 4 завдання 3 перейшло в стан готовності, де термін виконання раніше завдання 2 . Виявилося, що питання 3 пропустило термін. Тому алгоритм EDF без зміщення не є оптимальним.

На рисунку 1. 4 зображено розклад, складений EDF–плануванням з можливістю витіснення задач.

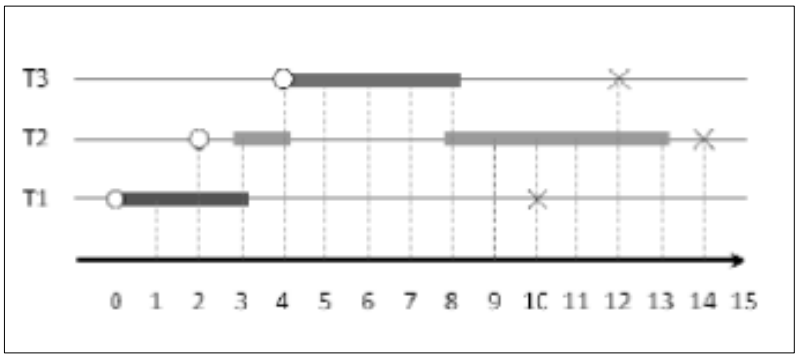


Рисунок 1.4 — Можливі розклади, підготовлені плануванням EDF із зміщенням завдань.

Як видно, у цьому випадку всі завдання виконуються до критичного терміну. У момент 2 завдання 2 готове, але воно замінює завдання 1, оскільки завдання 2 має більш пізній термін, ніж завдання 1. Місія 2 отримує контроль лише в момент 3. Однак у момент 4 з’являється завдання 3 і замінює завдання 2, оскільки завдання 3 має більш ранній термін, ніж завдання 2. У момент 8, коли завдання 3 завершує свою роботу, керівництво знову отримує завдання 2.

Підсумовуючи, не слід робити висновок, що EDF без зміщення гірше, ніж EDF зі зміщенням, оскільки докази оптимальності останнього наводяться в припущенні, що фактичне зміщення (і перемикання контексту) завдання не потребує жодного часу, який не є реальним.

Динамічне планування із статичними пріоритетами. Існує два поширені способи встановлення пріоритетів:

— RMS (rate monotonic scheduling);

— DMS (deadline monotonic scheduling).

Правила визначення пріоритетів у RMS-пріоритеті такі: чим коротший період виконання завдання, тим вище пріоритет. Іншими словами, чим частіше завдання переходить у стан готовності, тим вище його пріоритет. У DMS-пріоритеті алгоритм визначається дещо інакше: чим менше відносний термін виконання завдання, тим вище його пріоритет.

На рисунку 1. 5 наведено приклад розкладу, підготовленого плануванням RMS для синхронної системи трьох періодичних задач з наступними параметрами: T_1 (3;0,5); T_2 (4;1); T_3 (6;2).

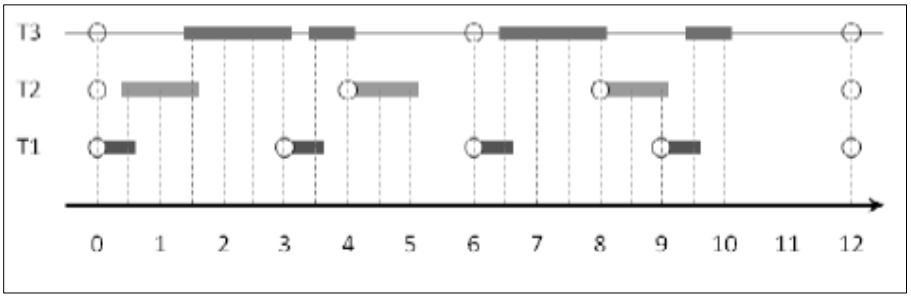


Рисунок 1.5 — Розклад складений RMS-плануванням

Відносний крайній термін дорівнює періоду.

Відповідно до вищезазначених правил, перше завдання має найвищий пріоритет, а третє — найнижчий. У момент 0 у нас є 3 готові завдання, і керівництво отримує перше завдання. На час 0,5 готові два завдання (2 і 3), керівництво отримує друге. У момент 3 перше завдання знову готове, тому воно замінює третє завдання. У момент 3.5 контроль знову отримує третій квест як річ. У момент 4.0 тільки

завдання 2 готове і бере контроль. У момент 6 два завдання (1 і 3) готові і контроль передається до першого. Тоді третє завдання працює, але на момент 8 замінюється другим завданням. У момент 9 перше завдання знову готове, воно виконується, і, нарешті, в момент 9.5, третє завдання керується. Потім все повторюється спочатку (за умови незмінної кількості завдань у плані).

На рисунку 1. 6 наведено приклад розкладу, складеного DMS-плануванням для наступної системи задач із наступними параметрами :T_1 (3;0,5); T_2 (4; 1); T_3 (6; 2).

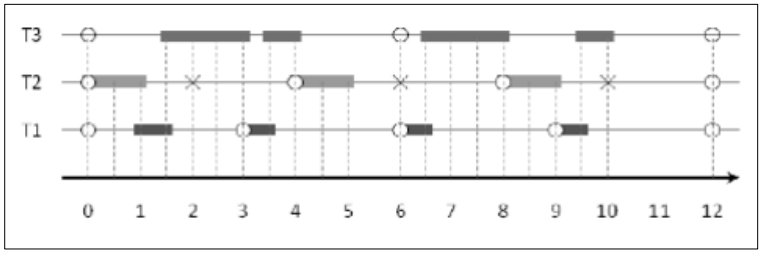


Рисунок 1.6 — Розклад складений DMS-плануванням

Єдина відмінність цієї системи від попередньої полягає в тому, що в цій задачі відносний член не дорівнює періоду, а менше ніж у два рази. Згідно з правилами пріоритету, друге завдання матиме найвищий пріоритет, а найменший – третє.

1.2 Основні параметри діаграми Ґанта, що використовуються для вивчення дисциплін

Гістограма (діаграма Ганта) — це стовпчаста діаграма (гістограма), яка використовується для ілюстрації плану роботи або плану проекту.

Діаграма Ганта є одним із інструментів для управління та планування проектів.

Перший формат діаграм був розроблений Генрі Гантом у 1910 році. Тоді вони складалися на папері. З розвитком комп’ютерів у 1980-х роках діаграми Ганта стали більш складними та детальними. До сьогодні вони залишаються одним з найпопулярніших інструментів планування проектів.

Діаграма Ганта складається з сегментів, розташованих у горизонтальній часовій шкалі. Кожен сегмент має окреме завдання або підзавдання. Зазвичай він складається з двох частин: ліворуч списку завдань, а праворуч — часової шкали зі смугами, що зображують роботу. Заплановані завдання, підзавдання та компоненти розміщені вертикально. Початок, кінець і довжина сегментів на часовій шкалі відповідають початку, кінця та часу виконання завдання. Деякі діаграми Ганта показують зв'язок між контрольними точками і завданнями [4]

Приклад діаграми Ґанта зображено на рисунку 1.7.

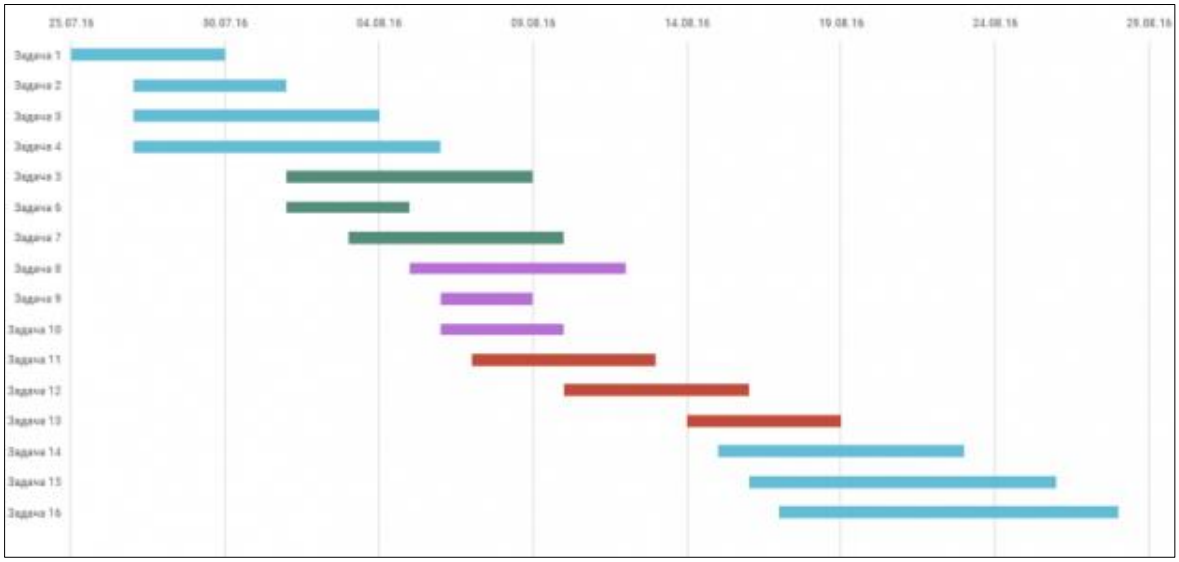


Рисунок 1.7 — Діаграма Ґанта

Одним з основних елементів діаграми Ганта є «віхи» — маркери важливих моментів у робочому процесі та спільна межа для багатьох завдань. Віхи необхідні, щоб проілюструвати необхідність послідовності та синхронізації під час виконання різних завдань. Віхи та інші межі на діаграмі Ганта не є календарними датами. Порушення етапів може призвести до зриву всього проекту Отже, діаграма Ганта не є робочою діаграмою. Крім того, діаграми Ганта не відображають важливість, ресурсомісткість і характер роботи. Для великих проектів діаграми Ганта стають дуже складними і втрачають чіткість.

Є дві основні причини, чому діаграми Ганта настільки популярні в управлінні проектами. Це спрощує створення складних планів, особливо тих, що включають

кілька команд і змінюють терміни. Діаграми Ганта допомагають командам вчасно планувати роботу та правильно розподіляти ресурси.

Головна перевага цієї діаграми - її універсальність. Основне питання полягає в тому, наскільки детально ви хочете вказати процеси та завдання.

Діаграми Ганта дозволяють переглянути проект збоку і оцінити обсяг і терміни виконання завдань. Чим більше завдань і підзадач, тим гірша візуалізація.

Головна проблема, з якою стикається практично кожен, хто користувався діаграмами Ганта, — це терміни виконання завдань. На щастя, було винайдено багато простих у використанні та простих інструментів управління проектами, які використовують діаграми Ганта як основу для візуалізації проблем.

За допомогою діаграм Ганта ви можете відстежувати та переглядати:

— які завдання входять до проекту;

— тривалість задач: час початку та закінчення;

— хто відповідає за кожне конкретне завдання;

— дати початку та завершення проекту;

— скільки часу займає кожне завдання.

Перевагою діаграми Ганта є графічний огляд і широкий діапазон використання. Основною перевагою графічного огляду є надання користувачам графічного та інтуїтивно зрозумілого дисплея. Діаграма складається з горизонтальних різнокольорових смуг, кожна з яких відповідає за роботу якихось людей або певну діяльність. Поширеним є те, що календар можна використовувати для планування практично будь-яких завдань, що робить його відомим рішенням загального призначення.

1.3 Порівняльний аналіз існуючого сучасного програмного забезпечення, які використовують діаграму Ґанта для планування

Порівняння програм, у яких використовується діаграма Ґанта, наведено у таблиці 1.3.

Таблиця 1.3 — Порівняльна характеристика аналогів із додатком, що розробляється

Характеристика	Microsoft Excel	Microsoft PowerPoint	Microsoft Visio	LibreOffice	GanttProject	TeamGantt	Додаток, що розробляється
Статичне планування	+	+	+	+	+	+	+
Вказування термінів	+	+	+	+	+	+	+
Редагування термінів	-	-	-	-	+	-	+
Шкала виконання	+	+	+	-	+	-	+
Повний доступ	+	+	+	+	-	-	+

У Microsoft Excel можна шукати різні моделі діаграм Ганта, ввівши діаграми Ганта у вікні пошуку на верхній панелі, де потрібно вибрати одну з п’яти моделей, запропонованих програмою. Більшість графічних робіт виконано, залишилося лише змінити клітинки з назвами проекту, змінити назви різних видів діяльності, заповнити заплановані дати початку та завершення, ввести дати початку та завершення та відсоток завершення. Кожна зміна в комірці відповідає модифікації праворуч, яка насправді є діаграмою Ганта.

У Microsoft PowerPoint з відкритою діаграмою Ганта ви можете змінити назву та вставити різні елементи діаграми, щоб побачити назви різних видів діяльності, тривалість різних ліній діаграми та наявність або відсутність жовтих ромбів. На другому слайді зазвичай є легенди діаграм, які допомагають зрозуміти введені дані.

У Microsoft Visio під час першого запуску моделі вам буде запропоновано ввести дані, що характеризують сценарій, наприклад кількість подій, дату початку, дату завершення проекту та очікувану тривалість. Після початкової компіляції вам потрібно натиснути «ОК», щоб створити фактичну діаграму Ганта, щоб внести всі

необхідні зміни, вам потрібно клацнути будь-яку точку діаграми або скористатися зручною боковою панеллю з основними кнопками для керування такими діаграми.

LibreOffice дозволяє змінити всі клітинки, які потрібно заповнити, і вставити назву діяльності, тривалість проекту та ефективну тривалість, щоб змінити рядок у правій частині електронної таблиці (справжня діаграма Ганта). Ця програма розроблена для тих, хто хоче генерувати дуже прості та лінійні діаграми Ганта.

У Microsoft Visio під час першого запуску моделі вам буде запропоновано ввести дані, що характеризують сценарій, наприклад кількість подій, дату початку, дату завершення проекту та очікувану тривалість. Після початкової компіляції вам потрібно натиснути «ОК», щоб створити фактичну діаграму Ганта; щоб внести всі необхідні зміни, вам потрібно клацнути будь-яку точку діаграми або скористатися зручною боковою панеллю з основними кнопками для керування такими діаграми. Також проект можна налаштувати так, щоб під час роботи всі зміни та оновлення, які були внесені, можна було відстежувати [5].

TeamGantt — це веб-додаток, який можна безкоштовно використовувати для керування проектами до трьох осіб. Далі є менш обмежувальні абонентські плани. На веб-сайті TeamGantt з’являється інтерфейс, подібний до MS Project, і ви можете відразу ж почати створювати файли керування проектами.

2 ОСОБЛИВОСТІ СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПЛАНУВАННЯ ЗА РОЗКЛАДОМ НА ОСНОВІ ДІАГРАМИ

ҐАНТА

2.1 Визначення параметрів діаграми Ґанта для планування

Діаграма Ганта використовується для представлення поточного стану роботи: частина прямокутника, що відповідає завданню, заштрихована, що вказує на відсоток виконаного завдання, а вертикальна лінія відображається, щоб позначити роботу, виконану «сьогодні».

Зазвичай діаграми Ганта використовуються з таблицею зі списком робіт, де рядки відповідають за одне завдання, як показано, а кожен стовпець містить додаткову інформацію про завдання. Діаграми Ганта дозволяють чітко і чітко відобразити компоненти проекту та розділити їх на менші завдання для полегшення керування. Сформовані завдання розміщуються вздовж шкали часу діаграми Ганта, а потім додаються залежності між завданнями, виконавцями та контрольними точками.

Взаємозалежність між завданнями гарантує, що нове завдання може бути виконане лише після завершення попереднього завдання. Якщо є затримка у виконанні будь-якого завдання, пов’язані завдання автоматично передаються. Це особливо корисно, якщо ви плануєте працювати з кількома командами.

Для створення діаграм Ганто, спочатку потрібно створити таблицю і ввести в неї потрібні показники. Це можна зробити де завгодно: навіть на аркуші паперу або в додатку, який будує діаграму. Для заповнення форми потрібні три типи даних: назва завдання, дата початку завдання та запланована дата завершення. Щоб вказати фактичну дату завершення завдання, ці слова враховують умови та ресурси, необхідні для створення діаграми Ганта.

У багатьох проектах важко досягти ідеальної послідовності процесів, тому, коли вони починають накладатися, діаграма Ганта змінюється. Щоб створити кваліфіковану діаграму Ганта, потрібно дотримуватися деяких правил.

Важливо розуміти основну важливість проекту, розуміти, яких дій переходу необхідно досягти і що для цього потрібно зробити.

Щоб сформувати час, необхідний для виконання завдання, його необхідно ґрунтувати на суб'єктивних оцінках або статистиці того, хто буде виконувати завдання. Однак навіть розгорнутої статистики недостатньо, щоб точно встановити терміни, і якщо проект знаходиться в чітко спланованій зоні виробництва, є лише 30% шанс, що всі завдання будуть виконані вчасно. Через постійні зміни в часі користувачі змушені постійно коригувати зовнішній вигляд діаграми. Чим більше ви внесете коригувань, тим менше графік буде відповідати початковому плану.

Основне призначення діаграми — показати залежності виконуваної роботи. Генрі Гант дійшов висновку, що для кожного співробітника необхідно створити діаграми, а результати його роботи повинні бути прикладом для інших співробітників. Перша діаграма Ганта була створена і побудована, щоб пояснити, як відбувається процес передачі завдань від одного співробітника до іншого і скільки часу він займає. В результаті такі схеми стали використовуватися не тільки робітниками і простими людьми, але і в розважальних закладах, військових заходах, організації різноманітних процесів.

2.2 Аналіз і особливості визначення термінів навчання на основі розкладу

Час – унікальний ресурс. Його не можна накопичувати як сировину чи гроші. Хочемо ми того чи ні, але ми повинні використовувати його за фіксованою ціною — 60 секунд за хвилину. Час не можна вмикати чи вимкнути, відшкодувати чи замінити. Час є найбільш жорстким і негнучким елементом людського існування. Але ви можете проаналізувати, як він використовується. Його можна використовувати ефективно або неефективно, як і будь-який інший ресурс.

Тайм-менеджмент — це практика або процес навчання свідомого контролю часу, витраченого на певну діяльність, що, зокрема, може підвищити ефективність і продуктивність.

На основі розкладу для визначення умов навчання ви можете досягти власної траєкторії навчання, зосередившись на предметах, які відповідають вашим особистим інтересам. Тому оптимізація розкладу зменшить час, що витрачається на

різні дисципліни щодня, і виділить більше часу на засвоєння дослідницьких тем дисциплін. При цьому важливо не перевикористовувати відведений час на теми з предметів, що не вивчаються на вступному рівні відповідно до навчальної програми.

Основним завданням складання предметного розкладу є планування та забезпечення впорядкованого вивчення всіх предметів: взаємозв'язку між ними, правильної послідовності та чергування форм предметної навчальної роботи.

При організації дисциплінарної програми за будь-якою системою необхідно враховувати наступні умови:

— забезпечити максимально рівномірне навантаження;

— забезпечте гнучкість планування навантаження.

Для початку, краще приділяти менше часу вивченню предметів і поступово збільшувати час на вивчення основних тем кожного предмета. Важливо знайти баланс, щоб не було занадто мало чи занадто.

Підготуйте щоденний розклад предметів, ви повинні правильно розподілити свій час на вивчення тем кожного предмета. Менше часу слід витрачати на вступні та підсумкові теми дисциплін, оскільки більшу частину часу слід витрачати на дослідження основних і найважливіших тем, які можуть забезпечити найбільше знання та розуміння кожної дисципліни [6].

Складання дедлайнів для навчальних предметів вручну на основі розкладів часто займає багато часу. Через надзвичайну складність врахування всіх обмежень остаточне рішення може бути не задовільним. Тому велике значення надається автоматизації планування. У будь-якому випадку остаточний термін завершення визначається в період планування. Планування дає можливість відкласти на більш пізній термін, якщо вивчення предмета не може бути завершено в обумовлений час.

Існує багато класичних підходів до вирішення завдань планування. Хоча це виглядає просто, запуск методу моделювання та алгоритму затінення графіка дуже ефективний для складання невеликих розкладів. При реалізації алгоритму, заснованого на принципах імітаційного моделювання, можливості використання

розробленої системи обмежені, а внутрішні зміни невеликі, що вимагають істотних змін в алгоритмі.

2.3 Аналіз методів підвищення ефективності розкладу для вивчення на основі діаграми Ґанта

Діаграми Ганта корисні для планування, оскільки вони спрощують керування ними, зберігають початковий розмір і допомагають успішно завершити курс.

За допомогою програми для планування, яку легко розробити, ви можете створювати діаграми Ганта навіть без досвіду. Кожен користувач програми може створити розклад для кожної теми. Завдяки цій можливості навчання стає ефективнішим. Однією з найбільших перешкод для будь-якого плану є неконтрольоване зростання його розміру, коли ви не можете контролювати або регулювати свій час і ресурси.

Якщо можливо, завжди слід пам’ятати про терміни виконання, кожен предмет слід розділити на розділи, після чого кожному предмету слід призначити окремий термін. Щоб правильно ранжувати завдання, потрібно дотримуватись правила: «Чим ближче кінцевий термін для будь-якої теми, тим вищий пріоритет дослідження». Важливо зауважити, що те, що до кінцевого терміну залишається багато часу, ще не означає цього може бути виконано у встановлені терміни. Завершіть дослідження у встановлені терміни. Вивчення деяких тем може зажадати більше зусиль, тому краще виділити деякий час, встановивши власні терміни — це допоможе вам не пропустити заплановану дату завершення.

Оцініть, скільки часу знадобиться на детальне вивчення кожного, і введіть дані в додаток, а потім розрахуйте загальний час для вивчення всієї теми в повному обсязі.

У програмі для розробки ви можете використовувати діаграму Ганта, щоб зв’язати кілька тем за терміном виконання. Якщо дослідження на цю тему відбувається швидше або повільніше, ніж планувалося, терміни дослідження наступної теми автоматично змінюються. Якщо користувач виконує обробку

швидше, ніж запланований час навчання, весь час, що залишився, буде перенесено на наступну тему. І навпаки, якщо користувач не дотримається терміну, весь додатковий час, який він витрачає на дослідження обраної теми, буде використано наступного разу.

У розробленому додатку є функція, за допомогою якої після завершення будь-якого предмета ви можете намалювати діаграму Ганта, яка показує запланований (показаний червоним кольором) і фактичний (показаний синім) час, витрачений на вивчення кожного з цих предметів.

Автоматична зміна термінів у заявках сприяє швидкому прийняттю рішень і плануванню, що підвищить ефективність вивчення кожного предмета.

3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ВЗНАЧЕННЯ ТЕРМІНІВ ВИНАННЯ ЕТАПІВ БАКАЛАВРСЬКИХ РОБІТ НА ОСНОВІ ДІАГРАМИ ҐАНТА

3.1 Інтегроване середовище розробки IntelliJ IDEA

Коли справа доходить до вибору середовища розробки для створення Java—додатків, є кілька варіантів — Eclipse, NetBeans і IntelliJ IDEA Community Edition. Усі ці середовища розробки є відкритими.

Переваги використання цих середовищ розробки:

— безкоштовне розповсюдження;

— можливість поліпшення IDE;

— можливість додавання нових функцій і відповідно нових можливотей.

Чим більше людей хочуть покращити, тим легше тестувати нові функції, тому середовище розробки містить менше помилок.

На сьогоднішній день популярними є Eclipse і IntelliJ IDEA. Загалом усі вони мають приблизно однакову функціональність, і важко оцінити, що краще.

Стандартні функції наведених вище IDE:

— підсвічування синтаксису;

— компіляція коду;

— можливость рефакторингу;

— автогенерація коду;

— інтеграція збібліотеками і програмними каркасами;

— перевірка помилок;

— система підказок;

— автодоповнення;

— налагоджувач коду .

Завжди точаться суперечки про те, що краще за Eclipse та IntelliJ IDEA. IDE функціонують приблизно так само, вибір однієї над іншою є питанням особистих уподобань, але ми все одно спробуємо визначити, яке середовище найкраще для написання Java.

IntelliJ IDEA — це інтегроване середовище розробки програмного забезпечення для кількох мов програмування. Видання Community Edition IntelliJ IDEA підтримує інструменти тестування TestNG і JUnit, CVS, Subversion, системи контролю версій Mercurial і Git, інструменти компіляції Maven і Ant, мови програмування Java, Java ME, Scala, Clojure і Groovy.

Середовище включає в себе модуль графічного інтерфейсу візуального дизайну Swing UI Designer, редактор XML, редактор регулярних виразів, систему перевірки коду, систему моніторингу завдань і надбудови для імпорту та експорту проектів з Eclipse.

Eclipse — це інтегроване середовище розробки для програмного забезпечення для певної платформи. Основною перевагою використання Eclipse є готові плагіни та модулі, які легко встановлювати та оновлювати. З їх допомогою можна налаштувати інтерфейс Eclipse під конкретного користувача, встановити підтримку необхідних мов програмування, налаштувати власний редактор для кожного типу файлів, налаштувати тип налагодження проекту.

IntelliJ IDEA має «розумний» режим автозаповнення, який підтримує додавання імен класів і надання коду, навіть якщо ви вводите окремий ідентифікатор із середини конструкції. Тобто IDEA надає лише значущі параметри, тобто варіанти, які підходять для даного методу чи класу, тоді як Eclipse надає всі можливі варіанти, не перевіряючи їх придатність у цьому конкретному випадку.

Eclipse, і IDEA підтримують XML. Але IDEA редагує XML лише як текст, тоді як Eclipse дозволяє редагувати його як структуру та текст. У разі обробки великих файлів уповільнення робить редагування XML в IDEA майже неможливим, а Eclipse вдається обробляти великі файли швидше.

Візуальний редактор форм у IDEA працює швидше, ніж Eclipse. Тому є просте пояснення: Eclipse зберігає форму безпосередньо в коді, IDEA — проміжний файл xml.

Треба сказати, що обидві системи побудовані на додаткових модулях. Але Eclipse — це оболонка для модулів, а IDEA — це передусім середовище розробки

Java. Наприклад, в Eclipse можна встановити такі модулі: для Java, J2ME, C/C++, Perl, PHP, Python, LaTeX, SVN Team Provider тощо. IDEA має плагіни для Erlang, Scala, J2ME та Python.

Eclipse дозволяє відкривати кілька проектів в одному вікні, надаючи програмістам контроль над залежностями та відносинами. Intellij відкриває проект в іншому вікні, і це заплутує робочу область.

Intel Community Edition підтримує лише Java, Groovy і Scala. Однак, якщо ви плануєте створити сервер Python, наприклад, використовуючи Ajax і HTML для підключення до веб – сервера Java або іншої комбінації мов, Eclipse — найкращий вибір.

Хоча Eclipse має можливість додавати плагіни, такі як CheckStyle, автозаповнення IDEA за замовчуванням працює швидше та краще.

Чим більше плагінів встановлено в IDE, тим довше він завантажується і тим більше пам’яті він займає. Однак Eclipse обробляє та завантажує великі проекти швидше, ніж IDEA. Зазвичай проекти відкриваються швидше в Eclipse, оскільки Intellij індексує весь проект під час запуску, але подальша робота над проектом відбувається швидше та зручніше в середовищі Idea.

Обидві IDE мають SVN/GIT/GitHub та інші плагіни. Але плагіни в IDEA надійніші, мають кращий графічний інтерфейс і зручніші у використанні.

Багато розширень доповнюють середовище Eclipse за допомогою менеджерів баз даних, серверів додатків тощо. Це такі плагіни, як:

— jOra, призначений для розробників Oracle і Oracle адміністраторів;

— плагін EclipseDatabase;

— графічна платформа Data Service;

— QuantumDB плагін та інші.

Intellij має вбудовану базу даних, тому вам не потрібно встановлювати додаткові плагіни.

IDEA — це потужне середовище розробки Java. Це краще, ніж Eclipse як Java IDE. Якщо ви використовуєте їх для різних цілей, наприклад, IDE для іншої мови

(C++, C, PHP, Perl, Ruby), або як платформу для створення настільних програм, вам слід вибрати Eclipse. В IDEA додаткові модулі не важливі. Все необхідне є в дистрибутиві.

3.2 Розробка алгоритму роботи додатку

Розробка програмного забезпечення IDEA — сфера програмування, яка швидко розвивається. Основне завдання розробника — створити зручний додаток для платформи IDEA, який буде працювати бездоганно і буде інстинктивно зрозумілим, вигідним і універсальним для користувачів програми.

Процес розробки програми IDEA складається з наступних етапів:

— підготовка технічних завдань;

— складання процесу та стадій функціонування;

— створення алгоритму роботи;

— програмування;

— створення інструкції щодо використання готових програми;

— будування архітектури;

— налагодження та тестування;

— підтримка та оновлення;

— оформлення документаці.

Після періоду відбору та побудови необхідної моделі початкових знань, природно, на основі термінів цієї моделі знайти її рішення. Основною метою тут є складання рішення шляхом створення алгоритму (проблеми) Рішення створюється з обмеженої кількості інструкцій, очевидно важливих і виконується за обмежений час з обмеженими обчислювальними витратами. Їм не заборонено втілювати в створюваному алгоритмі нескінченний відсоток разів, і вони можуть визначати відсоток повторень. Однак алгоритм має завершитися виконанням кінцевої кількості інструкцій, незалежно від вхідних даних.

Отже, можна зробити висновок, що розроблений алгоритм не повинен закінчуватися нескінченною кількістю циклових обчислень.

На рисунку 3.1 зображено граф переходів у розробленому додатку.

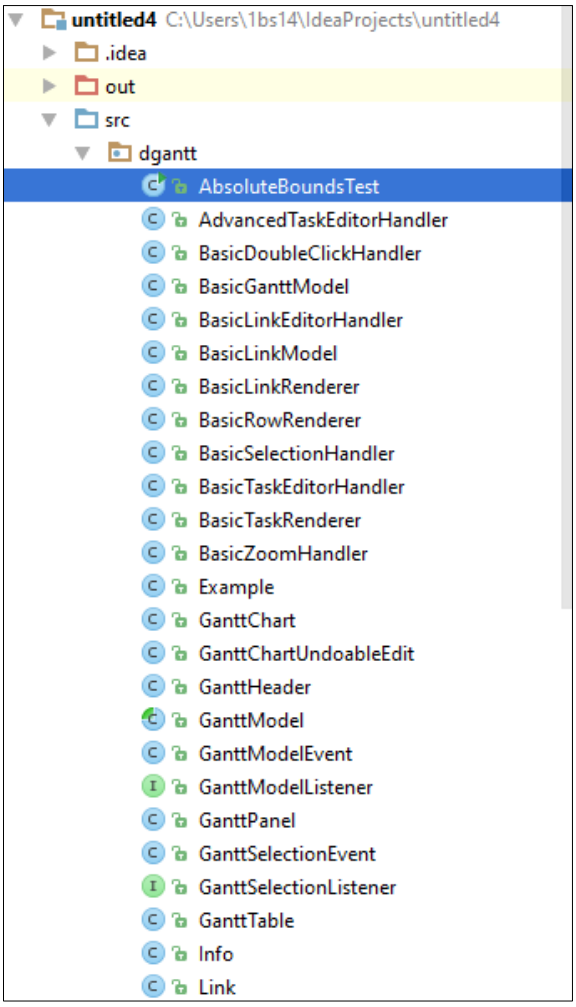


Рисунок 3.1 зображеня граф переходів у розробленому додатку.

Є й інший спосіб визначення алгоритму. Інструкції алгоритму повинні мати «однозначний сенс» і бути реалізовані з «обмеженими обчислювальними витратами». Звичайно, один користувач може зрозуміти частину, яка має «ясний сенс», і та сама частина може виглядати інакше для іншого. Те ж саме стосується концепції «кінечної вартості»: якщо ви чітко розумієте, що означає кожна інструкція, часто важко стверджувати, що інструкція закінчиться будь-якими вихідними даними [12].

Алгоритм — це механізм, який повинен не тільки гарантувати, що рішення знайдено, але й оптимальне рішення. Властивості правильного алгоритму композиції:

— обмежений час — це алгоритм повинен бути завершений в найкоротші терміни;

— коректність — це алгоритм мусить знаходити тільки вірні розв'язки;

— однозначність — це незалежно від того, яку кількість разів виконується алгоритм з однаковими вхідними даними, розв'язок має бути ідентичним;

— чіткість — це кожен етап алгоритму має бути чітко пояснений;

— скінченість — це алгоритм має містити обмежену кількість етапів.

Додаток працює за наступним алгоритмом. Після запуску програми користувач переходить на головну сторінку, і програма буде чекати команд користувача. Йому нададуть план захисту, готовність студентів та інформацію про розробника.

Після вибору студентів програма в окремому вікні створює діаграму Ганта, яка показує час для кожної теми есе, інакше діаграма Ганта не була б завершена. Після побудови діаграми Ганта програму закривають і випускають.

Алгоритм роботи додатку зображено на рисунку 3.2

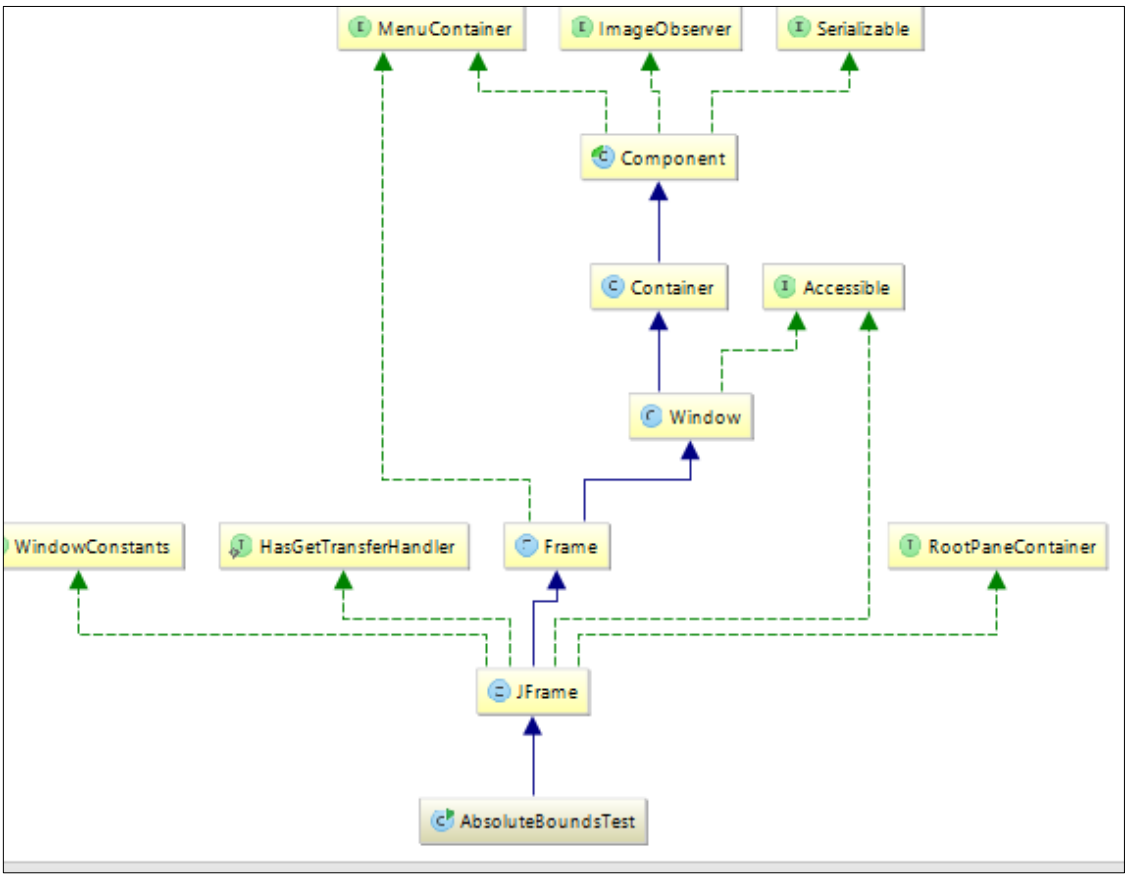


Рисунок 3.2 — Алгоритм роботи додатку

3.3 Розробка інтерфейсу додатку

Проектування додатків та розробка інтерфейсу є найважливішими етапами створення програмного продукту. Залежно від цієї роботи користувач буде сприймати додаток, чи сподобається він йому, чи буде додаток зручним у додатку, чи стане популярним.

Інтерфейс програми відіграє дуже важливу роль у процесі застосування, це механізм зв'язку між апаратним і програмним забезпеченням та центр взаємодії з користувачами.

Основною метою будь-якого додатка є забезпечення максимальної зручності та ефективності обробки інформації. Тому інтерфейс є найважливішою частиною будь-якої програми.

Дизайн програми має бути максимально простим і зрозумілим. Правильно розроблена програма має поєднувати три основні властивості. Це просто у використанні, тобто програма повинна мати інтуїтивно зрозумілий дизайн, інтегрувати та використовувати всі функції. Добре розроблений додаток міститиме функціональні можливості для використання кожної функції. Наступна особливість полягає в тому, що користувач повинен зацікавитися додатком. Найкращий спосіб змусити людей використовувати програму з багатьма функціями — додати користувачам елемент розваги та мотивації. Остання особливість — це корисність, оскільки найвищі оцінки мають ті програми, яким це дійсно потрібно. Під час розробки програми були дотримані всі перераховані вище правила та розроблено красивий та інтуїтивно зрозумілий інтерфейс [13].

Інтерфейс програми на основі діаграми Ганта вважає підготовку студентів до здачі бакалаврської роботи простою та красивою. На головній сторінці програми є назва програми та три кнопки «План захисту» та «Готовність студента» та «Інформація для розробника».

На рисунку 3.3 зображено вікно додатку «Головна сторінка додатку».

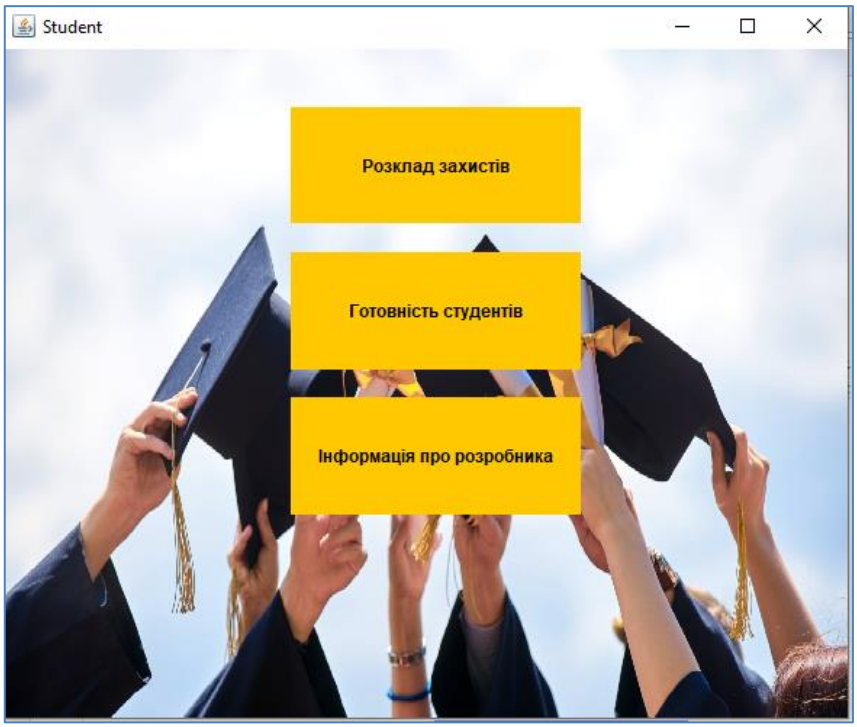
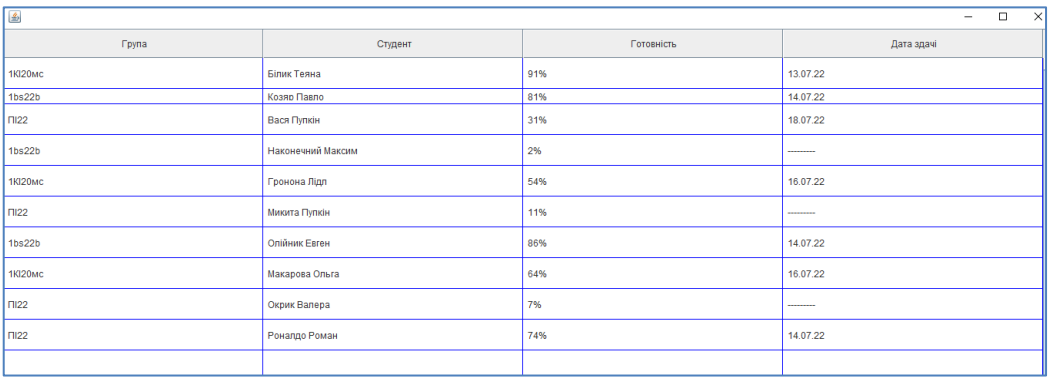


Рисунок 3.3— Вікно розробленого додатку «Головна сторінка додатку»

У всіх інших вікнах програми, крім домашньої сторінки, є кнопка «План захисту», яка дозволяє користувачеві перейти від будь-якого вікна до запланованого вікна, і кнопка «Підготовка учня», яка дозволяє користувачеві переглядати підготовку учнів. . На малюнку 3.4 показано вікно План захисту. Після завершення бакалаврської роботи користувачі можуть проаналізувати свій час, перейшовши у вікно програми, де зображено діаграму Ганта, що порівнює запланований час і фактично написаний на тему. На малюнку 3.5 показано вікно програми, що описує діаграму Ганта.



Група	Студент	Готовність	Дата здачі
1КІ20мс	Біпик Теяна	91%	13.07.22
1bs22b	Козяр Павло	81%	14.07.22
ПІ22	Вася Пупкін	31%	18.07.22
1bs22b	Наконечний Максим	2%	--------
1КІ20мс	Гронона Лідл	54%	16.07.22
ПІ22	Микита Пупкін	11%	--------
1bs22b	Опійник Евген	86%	14.07.22
1КІ20мс	Макарова Ольга	64%	16.07.22
ПІ22	Окрик Валера	7%	--------
ПІ22	Роналдо Роман	74%	14.07.22

Рисунок 3.4 — Вікно розробленого додатку, де користувач переглядатиме дату захисту

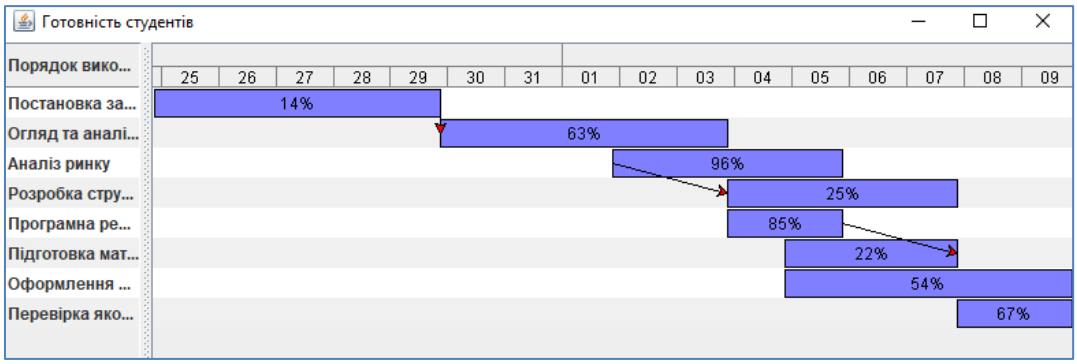


Рисунок 3.5 — Вікно розробленого додатку, яке зображує діаграму Ґанта готовності студента

3.4 Тестування роботи додатку

Тестування програмного забезпечення – це процес технічного дослідження, що виконується на запит замовника для визначення характеристик програми щодо параметрів використання продукту; процес аналізу або роботи програмного забезпечення для виявлення дефектів.

Тестування передбачає «аналіз» або «роботу» з розробки програми. Статичне тестування — це тестування, яке передбачає аналіз результатів розробки програмного забезпечення. Статичне тестування передбачає перегляд програм, контроль і перевірку додатків. Динамічне тестування — це тестування, яке передбачає роботу програмного продукту. Статичне та динамічне тестування доповнюють одне одного.

Тестування програмного коду – це процес виконання програмного коду, призначеного для виявлення наявних дефектів. Помилка – це частина програмного коду, яка при виконанні за певних умов призводить до несподіваної поведінки гаджета. Несподівана поведінка системи може призвести до збоїв в роботі, в цьому випадку в програмному коді є значні дефекти. Деякі помилки можуть викликати незначні проблеми, які не порушують систему, але створюють певні труднощі у її використанні. В даному випадку це середній або незначний дефект.

Мета застосування процедур тестування коду — мінімізувати кількість дефектів у кінцевому продукті. Тестування не гарантує, що в системному коді немає

дефектів. Однак у поєднанні з процесом верифікації та валідації, спрямованим на усунення невідповідностей та неповноти в проектній документації, добре організоване тестування гарантує, що система відповідає вимогам і працює відповідно до вимог за всіх передбачуваних обставин [14].

Запустивши додаток, користувачі можуть дізнатися, чи готовий певний студент до захисту бакалаврської роботи. Ми перевіряємо готовність студентів захищати ступінь бакалавра в розробленому додатку, і ми це перевіримо.У вікні студента виберемо групу«компютерна інженерія», студентку «Білк Тетяну Василівну». Вікно реєстрації додатку з введенними даними зображено на рисунку 3.6

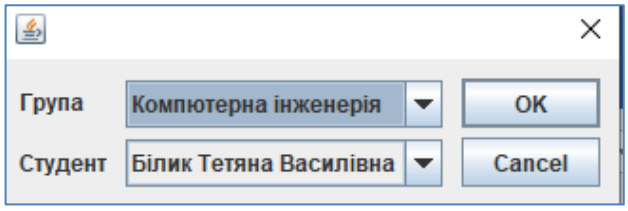


Рисунок 3.6— Вікно студеннта у додатку

Перевіримо, чи дійсно відбудеться обчислення готовності студента до захисту, яка зображена на рисунку 3.7

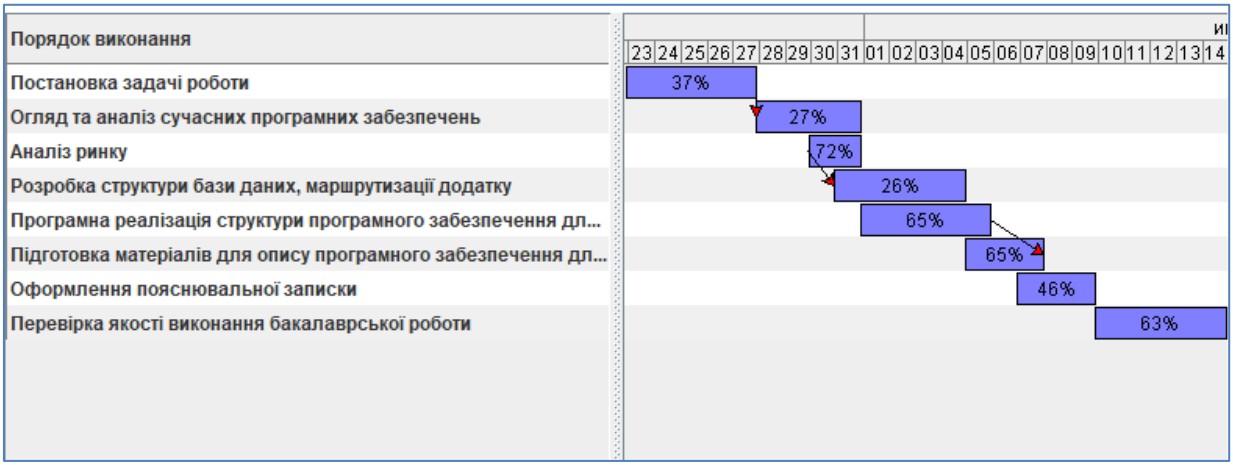


Рисунок 3.7— Вікно готовності студента до захисту додатку

3.5 Інструкція користувача

Зазначити, які ресурси необхідні для роботи програми, що необхідно зробити для запуску програми на виконання, дати чіткі вказівки, як необхідно відповідати на запити програми. Для запуску програми необхідно запустити файл AbsoluteBoundsTest.exe. Після запуску файлу з'являється заставка програми, а при натисканні будь-якої клавіші з'являється меню. У меню виберіть потрібний пункт, ввівши відповідне число, і натисніть Enter.

ВИСНОВКИ

У дипломній роботі було створено програмне забезпечення для врахування готовності у термінах написання дипломної роботи на основі діаграми Ґанта.

Програма Java, розроблена для ефективного планування часу студентів на підготовку до захисту дипломної роботи та прийняття рішень щодо підготовки в реальному часі.

Розглянуто існуючі методи та програмне забезпечення для планування за допомогою діаграм Ганта. Для цього ми розглядаємо типи пріоритетів, які використовуються для виконання робіт, аналізуємо основні параметри діаграм Ганта на предмет готовності наукової роботи, порівнюємо програми, що використовують діаграми Ганта, розглядаємо переваги розробки додатків.

Проаналізовано характеристики програмного забезпечення для створення з метою врахування змін у підготовці наукових робіт за діаграмою Ганта за планом. Для цього визначаються параметри плануючої діаграми Ганта, аналізуються визначення термінів навчання на основі розкладу та аналізуються методи підвищення ефективності навчання на основі діаграми Ганта.

Згодом було розроблено програмне забезпечення для обліку змін у часі вивчення предмета відповідно до графіка діаграми Ганта. Спочатку аналізується інтегроване середовище IntelliJ IDEA та вибирається мова програмування для розробки додатків. Потім розробили алгоритм розробки додатків, проаналізували розробку інтерфейсу та дизайн додатка, протестували роботу програми та написали посібник користувача для використання розробленого додатка.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Федотенко М. Разработка приложений. Первые шаги / М. Федотенко — Изд.: «Бином. Лаборатория знаний», 2016. — 335с.

2. Эрджиес К. Распределенные системы реального времени. Теория и практика К. Эрджиес — Изд.: «ДМК Пресс», 2020. — 382с.

3. Маккеон Ґ. Есенціалізм. Мистецтво визначати пріоритети / Ґ. Маккеон — Вид.: «Наш формат», 2021. — 224с.

4. Рассел Д. Диаграмма Ганта / Д. Рассел. — Изд.: «VSD», 2017. — 591 с.

5. Бучинський В.О. Програмне забезпечення для врахування змін у термінах вивчення дисциплін за розкладом на основі діаграми Ґанта. / В. О. Бучинський, А. В. Снігур. // Тези доповіді. L Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії. — Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2021/paper/view/12067.

6. Кноблаух Й. Управление временем. Второе издание / Й. Кноблаух, Х.Вёльте Москва: Изд. «Омега-Л», 2006. — 144 с.

7. Дарвин Я. Java. Сборник рецептов. Задачи и решения для разработчиков приложений / Я. Дарвин — Изд.: «Вильямс», 2016. — 768с.

8. Колисниченко Д. Программирование для java. Самоучитель. 3-е издание / Д. Колисниченко — Изд.: «БХВ-Петербург», 2020. — 288с.

9. Учебник[Електронний ресурс]. — 2019. — Режим доступу: https://coderlessons.com/tutorials/veb-razrabotka/izuchite-firebase/uchebnik-po-firebase.

10. Розробка мобільних додатків від А до Я: повний гайд [Електронний ресурс]. — 2021. — Режим доступу: https://dan-it.com.ua/uk/rozrobka-mobilnih-dodatkiv-vid-a-do-ja-povnij-gajd/.

11. Бурнет Э. Привет, Android! Разработка мобильных приложений / Э. Бурнет — Изд.: «Питер», 2016. — 256с.

12. Кнут Д. Искусство программирования. Т.1. Основные алгоритмы. — М.: Издательский дом «Вильямс», 2005. — 720 с.

13. Дейтел Х. Технологии программирования на Java 2. Книга 1: Графика, JavaBeans, интерфейс пользователя / Х. Дейтел — Изд.: «Бином-пресс», 2013 — 210с.

14. Котляров В. Основы тестирования программного обеспечения / В. Котляров, Т. Коликова — Санкт-Петербург: Изд.: «Бином. Лаборатория знаний», 2016. — 321 с.

ДОДАТОК А

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

ЗАТВЕРДЖУЮ
Завідувач кафедри _____
д.т.н., проф.
_____О. Д. Азаров
“___” _____ 2022р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання бакалаврської дипломної роботи
«Комп'ютерна підсистема для визначення термінів виконання етапів бакалаврських робіт на основі діаграми Ганта»
08-23.БДР.002.00.000 ПЗ

Науковий керівник к.т.н., доц. каф. ОТ
_____ Снігур А. В.
Студентки групи КІ–20мс
_____ Білик Т.В.

Вінниця 2022

1 Підстава для виконання бакалаврської дипломної роботи (БДР)

1.1 Підставою для виконання дипломної роботи є процес створення оптимальної структури захисту дипломних робіт та забезпечення безперервного планування захистів та врахуванню відсотків готовності роботи у режимі реального часу на основі діаграми Ґанта, наказ про затвердження теми дипломної роботи.

2 Мета і призначення БДР

2.1 Мета роботи—розробка додатку для ефективного планування часу готовності до захистів дипломних робіт;

2.2 Призначення розробки — виконання бакалаврської дипломної роботи із подальшим впровадженням розвитком.

3 Вихідні дані для виконання БДР: розробити Java–додаток, для врахування змін у термінах перевірки готовності бакалаврської роботи на основі діаграми Ґанта.

4 Технічні вимоги до виконання БДР: головна вимога — наявність Java – додатку та інформація для наповнення.

5 Етапи БДР та очікувані результати — етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 — Етапи БДР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		Початок	Кінець	
1	Аналіз сучасного стану досліджень	10.09.2022	25.09.2022	Огляд літературних джерел, задачі досліджень
2	Аналітичний огляд існуючих підходів та програмних засобів які використовують діаграму Ґанта для планування	26.09.2022	11.10.2022	Розділ 1
3	Особливості створення програмного забезпечення для перевірки готовності дипломної роботи на основі діаграми Ґанта	12.10.2022	31.10.2022	Розділ 2

Продовження таблиці А.1 — Етапи БДР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		Початок	Кінець	
4	Проектування та опис роботи програмного забезпечення для перевірки готовності дипломної роботи за розкладі на основі діаграми Ґанта	01.12.2022	03.01.2022	Розділ 3
5	Практична реалізація	01.12.2022	03.01.2022	Програмний продукт
6	Оформлення пояснювальної записки, презентації	04.01.2022	15.01.2022	Пояснювальна записка, презентація

6 Матеріали, що подаються до захисту БДР

До захисту подаються: пояснювальна записка БДР, презентація, протокол попереднього захисту БДР на кафедрі, відгук наукового керівника, відгук рецензента, анотації до БДР українською та іноземною мовами, нормоконтроль про відповідність оформлення БДР діючим вимогам.

7 Порядок контролю виконання та захисту БДР

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

8 Вимоги до оформлення БДР

При оформлюванні БДР використовуються:

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

— Методичні вказівки. Кафедра обчислювальної техніки 2022;

Технічне завдання до виконання отримала _____ Білик Т.В.

ДОДАТОК В

Лістинг функціонування головної сторінки додатку

```
ackage dgantt;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
public class AbsoluteBoundsTest extends JFrame {
    public AbsoluteBoundsTest(){
        super("Student");
        JPanel content = new JPanel();
        content.setLayout(null);
        JButton btnRozklad = new JButton("Розклад захистів");
        btnRozklad.setBounds(200,40,200,80);
        btnRozklad.setForeground(Color.BLACK);
        btnRozklad.setBackground(Color.ORANGE);
        btnRozklad.setBorderPainted(false);
        btnRozklad.setFocusPainted(false);
        JButton btnGotovnist = new JButton("Готовність студентів");
        btnGotovnist.setBounds(200,140,200,81);
        btnGotovnist.setForeground(Color.BLACK);
        btnGotovnist.setBackground(Color.ORANGE);
        btnGotovnist.setBorderPainted(false);
        btnGotovnist.setFocusPainted(false);
        btnRozklad.addActionListener(new ActionListener() {
                                public void actionPerformed(ActionEvent e) {
                                   //tf.setText("Welcome to Javatpoint.");
                                   new Tablestudent();
```

```
        btnGotovnist.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                //tf.setText("Welcome to Javatpoint.");
                Students dialog = new Students();
                dialog.pack();
                dialog.setVisible(true);
            }
        });
        JButton btnInfo = new JButton("Інформація про розробника");
        btnInfo.setBounds(200,240,200,81);
        btnInfo.setForeground(Color.BLACK);
        btnInfo.setBackground(Color.ORANGE);
        btnInfo.setBorderPainted(false);
        btnInfo.setFocusPainted(false);
        btnInfo.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                //tf.setText("Welcome to Javatpoint.");
                new Info();
            }
        });
        content.add(btnGotovnist);
        content.add(btnRozklad);
        content.add(btnInfo);
        BackgroundPanel bgp=new BackgroundPanel((new
ImageIcon("205837.jpg")).getImage());
        bgp.setBounds(0,0,600,500);
        content.add(bgp);
        setLayout(new GridBagLayout());
        setSize(600,500);
```

```
        //content.setBackground(Color.CYAN);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setContentPane(content);
    }
    class BackgroundPanel extends JPanel
    {
        Image im;
        public BackgroundPanel(Image im)
        {
            this.im=im;
            this.setOpaque(true);
        }
        //Draw the back ground.
        public void paintComponent(Graphics g)
        {
            super.paintComponents(g);
            g.drawImage(im,0,0,this.getWidth(),this.getHeight(),this);  } }
    public static void main(String[] args) {
        try {

UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());
        } catch (Throwable thrown) {
            thrown.printStackTrace();
        }
        AbsoluteBoundsTest abt = new AbsoluteBoundsTest();
        abt.setVisible(true);
    }
}
```

ДОДАТОК С

Лістинг функції AdvancedTaskEditorHandler

```
package dgantt;
import javax.swing.undo.UndoManager;
import javax.swing.undo.UndoableEdit;
import java.awt.*;
import java.awt.event.MouseEvent;
import java.awt.geom.Rectangle2D;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
public class AdvancedTaskEditorHandler extends BasicTaskEditorHandler {
  protected List<State> states;
  protected GanttChartUndoableEdit edit;
  protected UndoManager undoManager;
  protected boolean fireChangeDuringEdit = true;
  protected long minimumDuration = 1000 * 60 * 60 * 24;
  /**
   * Class constructor for an advanced task editor handler.
   *
   * @param chart the {@code GanttChart} connected to this listener.
   * @param undoManager the {@code UndoManager} where completed edits are
   *        committed
   */
  public AdvancedTaskEditorHandler(GanttChart chart,
      UndoManager undoManager) {
    super(chart);
    this.undoManager = undoManager;
```

```
  states = new ArrayList<State>();
}
/**
 * Returns {@code true} if the {@link GanttChart#fireChangeEvent()} method
 * is invoked as an edit is in progress; {@code false} otherwise.
 *
 * @return {@code true} if the {@link GanttChart#fireChangeEvent()} method
 *         is invoked as an edit is in progress; {@code false} otherwise
 */
public boolean isFireChangeDuringEdit() {
  return fireChangeDuringEdit;
}
/**
 * If {@code true}, the {@link GanttChart#fireChangeEvent()} method will be
 * invoked as an edit is in progress; {@code false} otherwise.
 *
 * @param fireChangeDuringEdit {@code true} if the
 *        {@link GanttChart#fireChangeEvent()} method is invoked as an edit
 *        is in progress; {@code false} otherwise
 */
public void setFireChangeDuringEdit(boolean fireChangeDuringEdit) {
  this.fireChangeDuringEdit = fireChangeDuringEdit;
}
@Override
public void mousePressed(MouseEvent e) {
  if (e.isConsumed()) {
    return;
  }
```

```
if (e.getButton() != MouseEvent.BUTTON1) {
  return;
}
Object selectedTask = chart.getTaskAtPoint(e.getPoint());
if (selectedTask == null) {
  if (!e.isControlDown()) {
    chart.clearSelection();
  }
} else {
  if (e.isControlDown()) {
    chart.toggleTaskSelection(selectedTask);
  } else if (!chart.isTaskSelected(selectedTask)) {
    chart.clearSelection();
    chart.selectTask(selectedTask);
  }
}
chart.repaint(chart.getVisibleRect());
if (e.isControlDown()) {
  return;
}
lastPoint = e.getPoint();
for (Object task : chart.getSelectedTasks()) {
  states.add(new State(task,
      chart.getTranslator().getStart(task),
      chart.getTranslator().getEnd(task),
      chart.getTranslator().getRow(task)));
}
mode = getMode(e);
if (mode == NONE) {
```

```
      return;}
    edit = new GanttChartUndoableEdit(GanttChartUndoableEdit.MOUSE, chart,
        undoManager);
    for (State state : states) {
      edit.addEditedTask(state.getTask());
    }
    edit.grabBeforeSnapshot();
    e.consume();
  }
  @Override
  public void mouseReleased(MouseEvent e) {
    if (e.isConsumed()) {
      return;
    }
    if (!e.getPoint().equals(lastPoint) && (edit != null)) {
      edit.grabAfterSnapshot();
      edit.commit();
    }
    edit = null;
    lastPoint = null;
    states.clear();
    chart.resize();
    chart.repaint();
    chart.fireChangeEvent();
    mouseMoved(e);
    e.consume();
  }
  @Override
  public void mouseDragged(MouseEvent e) {
```

```
if (e.isConsumed()) {
  return;
}
if (lastPoint == null) {
  return;
}
if (states.isEmpty()) {
  return;
}
Point point = e.getPoint();
long dx = chart.screenToCanonical(point.getX())
    - chart.screenToCanonical(lastPoint.getX());
for (int i = 0; i < states.size(); i++) {
  State state = states.get(i);
  Object task = state.getTask();
  long start = state.getStart();
  long end = state.getEnd();
  if (mode == RESIZE_START) {
    if (start + dx >= end - minimumDuration) {
      dx = end - start - minimumDuration;
    }
    start = start + dx;
  } else if (mode == RESIZE_END) {
    if (end + dx <= start + minimumDuration) {
      dx = start - end + minimumDuration; }
    end = end + dx;
  } else if ((mode == MOVE) || (mode == MULTIPLE)) {
    start = start + dx;
    end = end + dx; }
```

```
if ((chart.canonicalToScreen(start) < chart.getVisibleRect().getMinX())
    && (chart.canonicalToScreen(end) > chart.getVisibleRect().getMaxX())) {
  // do not scroll
} else {
  Rectangle2D bounds = chart.getTaskBounds(task);
  bounds.setFrame(chart.canonicalToScreen(start), bounds.getY(),
      chart.canonicalToScreen(end) - chart.canonicalToScreen(start),
      bounds.getHeight());
  chart.scrollRectToVisible(bounds.getBounds());
}
// snap dimension to days
Calendar cal = Calendar.getInstance();
if (mode != RESIZE_END) {
  cal.setTimeInMillis(start);
  cal.set(Calendar.HOUR_OF_DAY, 0);
  cal.set(Calendar.MINUTE, 0);
  cal.set(Calendar.SECOND, 0);
  cal.set(Calendar.MILLISECOND, 0);
  chart.getTranslator().setStart(task, cal.getTimeInMillis());  }
if (mode != RESIZE_START) {
  cal.setTimeInMillis(end);
  cal.set(Calendar.HOUR_OF_DAY, 0);
  cal.set(Calendar.MINUTE, 0);
  cal.set(Calendar.SECOND, 0);
  cal.set(Calendar.MILLISECOND, 0);
  chart.getTranslator().setEnd(task, cal.getTimeInMillis());   }
if (mode == MOVE) {
  Integer destinationRow = chart.getRow(point.getY());
  if (destinationRow != null) {
```

```
        int row = chart.getRow(lastPoint.getY())
              + (destinationRow - state.getRow());
        if (row != chart.getTranslator().getRow(task)) {
          chart.getTranslator().setRow(task, row);      }}      }
chart.resize();
chart.repaint();
if (isFireChangeDuringEdit()) {
   chart.fireChangeEvent();   }
e.consume();}
```

ДОДАТОК D

Лістинг функції BasicDoubleClickHandler

```
package dgantt;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.Rectangle2D;
public class BasicDoubleClickHandler extends MouseAdapter
implements ActionListener, FocusListener, ComponentListener {
  private final GanttChart chart;
  private Object task;
  private JTextField editor;
  public BasicDoubleClickHandler(GanttChart chart) {
    super();
    this.chart = chart;
  }
  public void startEdit(Object task) {
    this.task = task;
    int row = chart.getTranslator().getRow(task);
    Rectangle2D bounds = chart.getTaskBounds(task, row, 0, 0);
    editor = new JTextField(chart.getTranslator().getText(task));
    editor.addActionListener(this);
    editor.addFocusListener(this);
    editor.setBounds(bounds.getBounds());
    editor.setOpaque(true);
    editor.setHorizontalAlignment(JTextField.CENTER);
    chart.add(editor);
    chart.addComponentListener(this);
```

```
  editor.grabFocus();  }
/**
 * Invoked when the editor loses focus or editing is completed.
 */
public void stopEdit() {
  chart.remove(editor);
  chart.getTranslator().setText(task, editor.getText());
  chart.repaint(editor.getBounds());
  chart.removeComponentListener(this);
  editor = null;
  task = null;
  chart.fireChangeEvent();  }
@Override
public void mousePressed(MouseEvent e) {
  if ((editor != null) && (task != null)) {
    stopEdit();
    e.consume();
  }  }
@Override
public void mouseClicked(MouseEvent e) {
  if (e.isConsumed()) {
    return;}
  if ((e.getButton() == MouseEvent.BUTTON1) && (e.getClickCount() == 2)) {
    Object task = chart.getTaskAtPoint(e.getPoint());
    if (task != null) {
      startEdit(task);
      e.consume();
    }  } }
@Override
```

```
public void mouseMoved(MouseEvent e) {
  if (task != null) {
    chart.setCursor(Cursor.getDefaultCursor());
    e.consume();
  } }
@Override
public void actionPerformed(ActionEvent e) {
  if ((editor != null) && (task != null)) {
    stopEdit();   }}
@Override
public void focusGained(FocusEvent e) {
  //should never be called   }
@Override
public void focusLost(FocusEvent e) {
  if ((editor != null) && (task != null)) {
    stopEdit();   } }
@Override
public void componentHidden(ComponentEvent e) {
  if ((editor != null) && (task != null)) {
    stopEdit();    } }
@Override
public void componentMoved(ComponentEvent e) {
  if ((editor != null) && (task != null)) {
    Rectangle2D bounds = chart.getTaskBounds(task);
    editor.setBounds(bounds.getBounds());      } }
public void componentShown(ComponentEvent e) {
  if ((editor != null) && (task != null)) {
    stopEdit();
  } }}
```

ДОДАТОК Е

Лістинг функції BasicLinkModel

```
package dgantt;
import java.util.ArrayList;
import java.util.List;
/**
 * Basic {@link LinkModel} implementation.
 */
public class BasicLinkModel extends LinkModel {
  private List<Link> links;
  public BasicLinkModel() {
    links = new ArrayList<Link>();   }
  @Override
  public int getLinkCount() {
    return links.size(); }
  @Override
  public Link getLinkAt(int index) {
    return links.get(index);   }
  @Override
  public void addLink(Link link) {
    links.add(link);}
  @Override
  public void removeLink(Link link) {
    links.remove(link);  }
  public void addLink(Object from, Object to, LinkType type) {
    addLink(new Link(from, to, type));   }}
```

ДОДАТОК К

Лістинг функції Example

```
package dgantt;
import javax.swing.*;
import javax.swing.table.AbstractTableModel;
import javax.swing.undo.UndoManager;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.util.*;
import java.util.List;
public class Example {
  private static class Task {
    public String id;
    public String id1;
    public Date start;
    public Date end;
    public Task(String id,String id1, Date start, Date end) {
      super();
      this.id = id;
      this.id1 = id1;
      this.start = start;
      this.end = end;
   }}
  /**
   * An example translator using dates.
   */
  private static class TaskTranslator extends Translator {
```

```
private List<Task> modules;
public TaskTranslator(List<Task> modules) {
  super();
  this.modules = modules;}
@Override
public long getStart(Object task) {
  return ((Task)task).start.getTime();}
@Override
public long getEnd(Object task) {
  return ((Task)task).end.getTime();}
@Override
public String getText(Object task) {
  return ((Task)task).id1;}
@Override
public String getToolTipText(Object task) {
  return ((Task)task).id;}
@Override
public void setStart(Object task, long start) {
  Date startDate = new Date(start);
  ((Task)task).start = startDate;}
@Override
public void setEnd(Object task, long end) {
  Date endDate = new Date(end);
  ((Task)task).end = endDate;}
@Override
public void setText(Object task, String text) {
  ((Task)task).id1 = text;}
@Override
public int getRow(Object task) {
```

```
        return modules.indexOf(task);}    }
    void Example() {
      // generate the data
       String[] columnNames =new String[8];
      columnNames[0]="Постановка задачі роботи";
      columnNames[1]="Огляд та аналіз сучасних програмних забезпечень";
      columnNames[2]= "Аналіз ринку";
      columnNames[3]="Розробка структури бази даних, маршрутизації додатку";
      columnNames[4]="Програмна реалізація структури програмного забезпечення
для планування";
      columnNames[5]="Підготовка матеріалів для опису програмного забезпечення
для планування";
      columnNames[6]="Оформлення пояснювальної записки";
      columnNames[7]="Перевірка якості виконання бакалаврської роботи";
       Random random = new Random();
       String[] columnNames1 =new String[8];
       columnNames1[0]= String.valueOf(random.nextInt(100))+"%";
       columnNames1[1]= String.valueOf(random.nextInt(100))+"%";
       columnNames1[2]=String.valueOf(random.nextInt(100))+"%";
       columnNames1[3]= String.valueOf(random.nextInt(100))+"%";
       columnNames1[4]= String.valueOf(random.nextInt(100))+"%";
       columnNames1[5]= String.valueOf(random.nextInt(100))+"%";
       columnNames1[6]= String.valueOf(random.nextInt(100))+"%";
       columnNames1[7]= String.valueOf(random.nextInt(100))+"%";
      Calendar calendar = Calendar.getInstance();
      final List<Task> tasks = new ArrayList<Task>();
      for (int i = 0; i < 8; i++) {
        calendar.add(Calendar.DATE, -random.nextInt(5));
        Date start = calendar.getTime();
```

```
        calendar.add(Calendar.DATE, random.nextInt(4) + 2);
        Date end = calendar.getTime();
        tasks.add(new Task(columnNames[i],columnNames1[i], start, end)); }
    TaskTranslator translator = new TaskTranslator(tasks);
    GanttModel dataModel = new BasicGanttModel(tasks);
    // create links
    BasicLinkModel linkModel = new BasicLinkModel();
    linkModel.addLink(tasks.get(0), tasks.get(1), LinkType.FINISH_TO_START);
    linkModel.addLink(tasks.get(2), tasks.get(3), LinkType.START_TO_START);
    linkModel.addLink(tasks.get(4), tasks.get(5), LinkType.FINISH_TO_FINISH);
    // create the actual dgantt chart
    final GanttChart chart = new GanttChart(dataModel, translator, linkModel);
    // create a handler for box selection using the left-mouse button
    BasicSelectionHandler boxSelectionHandler = new BasicSelectionHandler(chart);
    chart.addMouseListener(boxSelectionHandler);
    chart.addMouseMotionListener(boxSelectionHandler);
    // create handler for manipulating links
    BasicLinkEditorHandler linkHandler = new BasicLinkEditorHandler(chart);
    chart.addMouseListener(linkHandler);
    chart.addMouseMotionListener(linkHandler);
    // create a handler for zooming using the right-mouse button
    BasicZoomHandler zoomHandler = new BasicZoomHandler(chart);
    chart.addMouseListener(zoomHandler);
    chart.addMouseMotionListener(zoomHandler);
    // create a handler for double-clicking a task to change its text
    BasicDoubleClickHandler doubleClickHandler = new
BasicDoubleClickHandler(chart);
    chart.addMouseListener(doubleClickHandler);
    chart.addMouseMotionListener(doubleClickHandler);
```

```
    // create a handler for moving, resizing and selecting tasks
    BasicTaskEditorHandler taskEditorHandler = new AdvancedTaskEditorHandler(chart,
new UndoManager());
    chart.addMouseListener(taskEditorHandler);
    chart.addMouseMotionListener(taskEditorHandler);
    // create the dgantt header
    final GanttHeader header = new GanttHeader(chart);
    // create a table listing the module id and description
    final GanttTable table = new GanttTable(chart, header,
        new AbstractTableModel() {
          private static final long serialVersionUID = 57211391812572479 21L;
          @Override
          public int getColumnCount() {
            return 1;    }
          @Override
          public int getRowCount() {
            return tasks.size();    }
          @Override
          public Object getValueAt(int rowIndex, int columnIndex) {
            return tasks.get(rowIndex).id;   }
          @Override
          public String getColumnName(int column) {
            return "Порядок виконання";}    });
    GanttPanel panel = new GanttPanel(chart, header, table);
    final JFrame frame = new JFrame("Готовність студентів");
    frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    frame.setSize(800, 600);
     frame.setLocationRelativeTo(null);
    JPanel buttonPane = new JPanel(new FlowLayout(FlowLayout.RIGHT));
```

```
buttonPane.add(new JButton(new AbstractAction("Close") {
   private static final long serialVersionUID = 38054269370562293 58L;
   @Override
   public void actionPerformed(ActionEvent e) {
      frame.dispose();
   }
}));
frame.getContentPane().add(panel, BorderLayout.CENTER);
frame.getContentPane().add(buttonPane, BorderLayout.SOUTH);
frame.setVisible(true);  }}
```

ДОДАТОК М

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА
НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Комп'ютерна підсистема для визначення термінів виконання етапів бакалавських робіт на основі діаграми Ганта.»

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 88,8% Схожість 11,2%

Аналіз звіту подібності (відмітити потрібне):

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

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Білик Т.В
(підпис) (прізвище, ініціали)

Керівник роботи _____ Снігур А.В.
(підпис) (прізвище, ініціали)