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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Підсистема коригування термінів вивчення обраних дисциплін системи дистанційного навчання

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав студент 1 курсу, групи 1КІ-21м
спеціальності 123 — Комп'ютерна інженерія
Бучинський В.О.
Керівник к.т.н., доц. каф. ОТ
Снігур А.В.
"17" 12 2022 р.
Опонент к.т.н., доц. каф. ПЗ
Коваленко О.О.
"19" 12 2022 р.

Допущено до захисту
д.т.н., проф. Азаров О.Д.
"20" 12 2022 р.

ВНТУ 2022

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

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

ЗАТВЕРДЖУЮ

Завідувач кафедри обчислювальної техніки

О.Д. Азаров
"15" 09 2022 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту **Бучинському Володимиру Олександровичу**

1 Тема роботи «Підсистема коригування термінів вивчення обраних дисциплін системи дистанційного навчання» керівник роботи к.т.н., доц. каф. ОТ Снігур А. В., затверджено наказом вищого навчального закладу від 15.09.2022 року № 205-А.

2 Строк подання студентом роботи 09.12.2022 року.

3 Вихідні дані до роботи: засоби – інтегроване середовище розробки Android Studio, база даних Firebase. Інтерфейс програми – підсистема для платформи Android, яка розроблена на мові програмування Java. Передбачено створення бази даних для збереження даних користувачів та збереження термінів вивчення тем обраних предметів для вивчення.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналітичний огляд програмних засобів та навчальних систем, які використовують діаграму Ганта для планування, моделювання та проектування підсистеми коригування термінів вивчення обраних дисциплін системи

термінів вивчення обраних дисциплін системи дистанційного навчання, економічна частина, висновки, література, додатки.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): технічне завдання, структурна схема, лістинг підсистеми коригування термінів вивчення обраних дисциплін системи дистанційного навчання, маршрутизація в підсистемі, база даних.

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
Розділ 1-3	к.т.н., доц. каф. ОТ Снігур А. В.		
Розділ 4	к.е.н., проф. каф. ЕПВМ Небава М. І.		

7 Дата видачі завдання 15.09.2022 року.

8 Календарний план виконання МКР приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Підпис
1	Постановка задачі	09.09.2022	Вик.
2	Огляд існуючих рішень	12-19.09.2022	Вик.
3	Проведення порівняльного аналізу аналогів	20-29.09.2022	Вик.
4	Розробка структурної схеми	02-07.10.2022	Вик.
5	Моделювання підсистеми	08-13.10.2022	Вик.
	Реалізація алгоритму роботи підсистеми.	14-21.10.2022	Вик.
5	Програмна реалізація структури підсистеми	22-30.10.2022	Вик.
6	Проектування підсистеми	02-10.11.2022	Вик.
7	Розрахунок економічної частини	11-18.11.2022	Вик.
8	Оформлення пояснювальної записки	19-27.11.2022	Вик.
9	Виконання магістерської кваліфікаційної роботи	09.09-01.12.2022	Вик.
10	Перевірка якості виконання магістерської кваліфікаційної роботи та усунення недоліків	01-10.12.2022	Вик.
11	Підписи супроводжувальних документів у керівника, опонента, нормоконтролера	13-20.12.2022	Вик.
12	Перевірка «антиплагіат»	12-19.12.2022	Вик.
13	Попередній захист	01.12.2022	Вик.

Студент _____ Бучинський В.О.

Керівник _____ к.т.н., доц. каф. ОТ Снігур А.В.

АНОТАЦІЯ

УДК: 378.018.43

Бучинський В.О. Підсистема коригування термінів вивчення обраних дисциплін системи дистанційного навчання. Магістерська кваліфікаційна робота зі спеціальності 123 — Комп’ютерна Інженерія, Вінниця: ВНТУ, 2022. 109с. На укр. мові.

Бібліогр.: 22 назв; рис.: 30; табл. 10.

У магістерській дипломній роботі розглядається питання розробки підсистеми коригування термінів вивчення обраних дисциплін системи дистанційного навчання. Під час виконання магістерської дипломної роботи було проаналізовано актуальність предметної області, розглянуто технології діаграми Ґанта, що використовуються для вивчення дисциплін, виконано огляд аналогів підсистеми, що розробляється. На основі отриманих даних було створено дану підсистему коригування термінів вивчення обраних дисциплін системи дистанційного навчання.

Було розроблено логічну структуру підсистеми. Для функціонування підсистеми було спроектовано базу даних. Проведено тестування роботи підсистеми.

Для розробки підсистеми використано мову програмування Java, програмне середовище Android Studio та базу даних Firebase.

Графічна частина складається з 5 плакатів із результатами моделювання.

Ключові слова: підсистема, android, розклад, планування, діаграма Ґанта, терміни, дисципліна, навчання.

ABSTRACT

Buchynskyi V.O. Subsystem for adjusting the terms of study of selected disciplines of the distance learning system. Master's thesis in specialty 123 — Computer Engineering, Vinnytsia: VNTU, 2022. — 109p.

In Ukrainian language. Bibliographer: 22 titles; fig.: 30; tabl. 10.

This master's thesis examines the issue of developing a subsystem for adjusting the terms of study of selected disciplines of the distance learning system. During the execution of the master's thesis, the relevance of the subject area was analyzed, Gantt chart technologies used for studying disciplines were considered, and analogues of the subsystem under development were reviewed. On the basis of the received data, this subsystem was created for adjusting the terms of study of selected disciplines of the distance learning system.

The logical structure of the subsystem was developed. A database was designed for the functioning of the subsystem. Subsystem operation was tested.

The Java programming language, the Android Studio software environment, and the Firebase database were used to develop the subsystem.

The graphical part consists of 5 posters with simulation results.

Key words: subsystem, android, schedule, planning, Gantt chart, deadlines, discipline, learning.

ЗМІСТ

ВСТУП

Актуальність теми дослідження полягає в тому, що розробка мобільних підсистем в епоху Інтернету — популярна послуга. Користувачів мобільних пристроїв стає все більше кожного дня. Зараз для вирішення різних завдань користувачам необхідна велика кількість мобільних програм, тому розвиток мобільних підсистем є актуальною проблемою. Актуальність розробки мобільних підсистем зростає із кожним днем. Велика кількість мобільних підсистем з'являється на різних мобільних платформах. Можливо, складно придумати щось нове, але розумна розробка в поєднанні з маркетингом зможе багато зробити на ринку підсистем Android.

Об'єктом дослідження є процес створення підсистеми коригування термінів вивчення обраних дисциплін системи дистанційного навчання за допомогою мови програмування Java.

Предметом дослідження є програмні засоби інтегрованого середовища розробки Android Studio, що використовуються для функціонування, підтримки підсистеми, та аналізу його складових.

Метою роботи є розробка підсистеми для ефективного планування часу вивчення обраних дисциплін.

Для досягнення цієї мети необхідно вирішити такі **задачі:**

— проаналізувати методи та технології діаграми Ґанта, що використовуються для вивчення дисциплін;

— провести аналіз аналогів програмного забезпечення, які використовують діаграму Ґанта для планування;

— розробити моделі врахування змін у термінах вивчення дисциплін та моделі навчання;

— реалізувати алгоритм роботи підсистеми;

— реалізувати структуру підсистеми коригування термінів вивчення обраних дисциплін системи дистанційного навчання;

— спроектувати підсистему коригування термінів вивчення обраних дисциплін системи дистанційного навчання;

— підключити підсистему до бази даних Firebase;

— протестувати розроблену підсистему.

Наукова новизна полягає у побудові моделі врахування змін у термінах вивчення дисциплін та моделі навчання, а також огляду математичних моделей визначення пріоритетів при виконанні різного роду робіт. Створені математичні моделі дозволяють значно спростити процес вивчення обраних дисциплін системи дистанційного навчання.

Практичне значення роботи полягає в тому, що запропоновано можливість ефективного безперервного планування робочого дня та прийняття рішень у режимі реального часу.

Апробація результатів роботи здійснена в доповіді на L науково-технічній конференції підрозділів Вінницького національного технічного університету. Зареєстровано патент на комп’ютерну програму «Підсистема коригування термінів вивчення обраних дисциплін системи дистанційного навчання» у державному підприємстві «Український інститут інтелектуальної власності».

Було зареєстровано патент на комп’ютерну програму «Підсистема коригування термінів вивчення обраних дисциплін системи дистанційного навчання» у державному підприємстві «Український інститут інтелектуальної власності».

Матеріали роботи доповідались та опубліковувались [1]:

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

1 АНАЛІТИЧНИЙ ОГЛЯД ПРОГРАМНИХ ЗАСОБІВ ТА НАВЧАЛЬНИХ СИСТЕМ, ЯКІ ВИКОРИСТОВУЮТЬ ДІАГРАМУ ҐАНТА ДЛЯ ПЛАНУВАННЯ

1.1 Актуальність та аналіз предметної області щодо особливостей організації планування

Планування — заздалегідь визначений набір питань для досягення поставленої мети. Планування — оптимізація розподілу ресурсів для досягнення поставлених цілей.

Слово «план» утворено від латинського слова «planum» — площина, рівне місце. Передусім слово планування використовувалось для позначення графіки, що зображує певну область у плоскому масштабі. Згодом його почали використовувати, щоб описати завдання, виконання яких передбачало взаємопов’язаний порядок обчислень, показників та дій.

Планування прогнозує знаходження відповідей на ключові питання:

— де знаходиться проблема — поточний стан?

— мета вирішення проблеми — куди вона рухається?

— як перемістити проблему з того місця де вона знаходиться зараз, туди, де вона хоче піти?

Планування є найважливішою функцією, тому що рішення, що приймаються під час реалізації, реалізовують усі інші функції, які пов’язані з плануванням.

Особливості планування:

— цілі затвердження;

— програмування;

— прогнозування;

— моделювання.

Основні процеси планування зображено на рисунку 1.1.

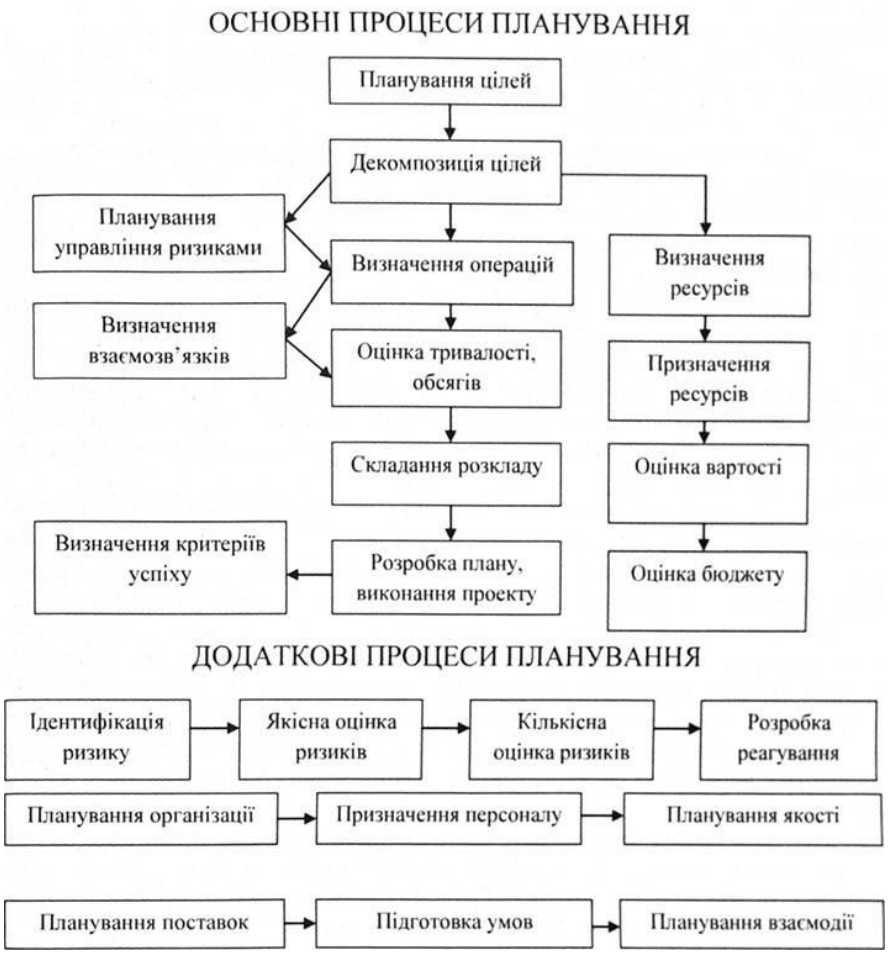

Рисунок 1.1 — Основні процеси планування

Планування створювали для того, щоб визначати цілі та способи їх реалізації відповідно до комплексу поставлених задач, та у визначенні різних методів, ресурсів та способів, які необхідні, щоб виконати обрані завдання, та встановити їх взаємозв'язки [2].

Метою планування є створення системи визначення змісту та певних напрямів дій щодо забезпечення довгострокового існування організації або підприємства.

Планування є основним механізмом та початком організації всього процесу досягнення цілей. До головних функцій організаційного менеджменту відносять організацію, планування, контроль, регулювання, облік, аналіз, стимулювання, координацію та активізацію, де саме планування відіграє головну роль, як зображено на рисунку 1.2.

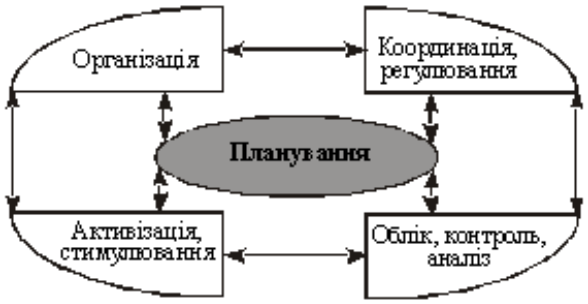

Рисунок 1.2 — Зв’язок планування з іншими функціями управління

Планування визначає недоліки ринку, та оскреслює дії, які необхідні щоб вирішити проблеми у майбутньому. Однак чітко передбачити всі параметри системи економіки, реалізуючи ці параметри, неможливо.

Мета плану — забезпечення ритмічного виробництва, економічного доходу та стабільного фінансового становища.

Найбільш загальна форма планування включає наступні етапи:

— визначення завдань, мети та цілей;

— розробка плану дій;

— визначити необхідні ресурси та їх джерела;

— визначення безпосередніх працівників та виконання поставленої програми.

Діаграма Ґанта користуються для відображення теперішнього етапу роботи: частина прямокутника, що відповідає завданню повинна бути заштрихованою, оскільки на ній позначається на скільки відсотків завдання є виконаним та зображено вертикальну лінію, що вказує на скільки робота є зробленою на «сьогодні».

Зазвичай, діаграмою Ґанта користуються з таблицею завдань, де рядок виконує одну задачу, що показане на діаграмі, де стовпчик показує інформацію задачі.

Діаграма Ґанта візуалізує компоненти проекту та розділяє їх на менші задачі, щоб ними було зручніше керувати. Створені завдання розташовуються на часовій

шкалі діаграми Ґанта, де потім до неї додаються зв’язки між завданнями та керівниками.

Зв’язки між задачами відповідають за те, щоб кожна нова задача виконувалась тільки після того, як попередня задача буде завершеною. Коли будь-яка задача затримується у виконанні, тоді пов’язані між собою задачі автоматично відкладаються. Це ефективно коли заплановано задачі разом не лише з одною командою.

Для створення діаграми Ґанта необхідно заповнити таблицю необхідними параметрами. Ви можете виконати це будь-де: у підсистемі створення діаграм Ґанта, або навіть на папері. Для заповнення таблиці потрібно вказати деякі типи даних: назва завдання, запланована дата початку та запланована дата завершення. Для того щоб спрогнозувати реальну дату завершення завдання, необхідно не забувати про ресурси та строки, які потрібні, щоб створити діаграму Ґанта.

Більшість проектів не можуть дійти до коректної залежності процесів, отже якщо проекти накладаються, тоді вигляд діаграми Ґанта змінюється. Щоб створити правильну діаграму Ґанта необхідно користуватися певними правилами, а саме необхідно розуміти мету проекту, які проміжні етапи повинні виконатися, розуміти що необхідно для цього зробити.

Щоб спланувати час, який потрібний щоб виконати поставлене завдання, необхідно базуватися на досвід користувача, який повинен виконати необхідне завдання, або проаналізувати статистику. Проте багато статичних даних буде недостатньо щоб ідеально встановити терміни, оскільки відсоток на те, що завдання виконаються в строки, дорівнює приблизно 25%, якщо проект виконується в чітко спланованому місці. Зрозуміло, що терміни зазнають змін, тому користувач повинен часто редагувати функціональність діаграм. Якщо виправлень буде зроблено багато, тоді діаграма буде виглядати зовсім інакше від початкового вигляду.

Призначення діаграми Ґанта — виявити зв’язки між зробленими роботами. Генрі Ґант заявив про необхідність створення діаграми кожному користувачеві,

оскільки це додасть стимулу кожному іншому користувачеві виконувати свою роботу краще у результаті. Для того, щоб зрозуміти як передавати завдання від одного користувача до іншого, і як багато часу потрібно витратити, щоб досягти поставленої мети, для цього й створювалися та будувалися перші діаграми Ґанта. В результаті, діаграму Ґанта можна було створювати, щоб організувати процес у закладах відпочинку, військових заходах та щоб організувати будь-який інший процес.

Діаграма Ґанта чудово підходить щоб розпланувати свій розклад, тому що вона спрощує користування ним, дозволяє зберігати розклад у початковій пропорції та допомагає не відставати від тем вивчення.

Створити свій власний розклад може абсолютно кожний користувач створеної підсистеми, оскільки в ньому доступна можливість створити власну діаграму Ґанта. Кожен користувач підсистеми може самостійно складати розклад для кожного власного створеного предмету. Завдяки цьому навчання є ефективним.

Проте на шляху будь-якого планування — неконтрольоване збільшення розміру графіку, тобто той момент, коли користувач не може встигати або редагувати втрачений час та ресурси.

Не забувайте про терміни виконання та намагайтеся розділити усі предмети для вивчення на декілька частин, і лише потім призначайте кожній створеній темі предмету для вивчення окремий термін виконання. Щоб правильно розділити задачі, необхідно слідувати правилу – «якщо наближається термін виконання у будь-якій темі створеного предмету, тоді пріоритет вивчення повинен збільшуватися». Також зауважте, якщо до термінів виконання ще достатньо часу, це не значить, що ви зможете вивчити створені теми протягом обумовленого часу. Оскільки вивчення більщ складних тем потребує більше зусиль, намагайтеся виділити більше часу, для цього встановіть власні терміни завершення — тоді ви не пропустититн дату, яку ви планували на завершення вивчення обраної теми.

Проаналізуйте як багато часу необхідно, щоб повністю вивчити кожну тему, після чого введіть дані у підсистему, та підрахуйте загальну тривалість часу для

повного вивчення всього предмету.

У підсистемі, що розробляється, за допомогою діаграми Ґанта ви можете поєднати теми обраного предмету для вивчення за терміном виконання. За умови, що вивчення обраної теми завершилося повільніше або швидше ніж було заплановано, в такому випадку автоматично зміняться строки вивчення наступної теми.

У випадку коли користувач підсистеми вивчив предмет швидше ніж було заплановано, тоді весь вільний час буде використано у наступній темі. Але й бувають випадки, коли користувач підсистеми не справився із запланованими термінами, тоді зайвий час, витрачений на вивчення обраної теми, буде віднято у наступної теми для вивчення.

Після того, як користувач завершив вивчення обраного предмету, створена підсистема має функцію, яка будує діаграму Ґанта, на якій буде зображено заплановані терміни та реальний час, який було витрачено на вивчення кожної теми предмету. Завдяки цьому підсистема допомагає швидко приймати рішення у плануванні графіка, тому буде підвищено ефективність у вивченні кожного створеного предмету користувача.

1.2 Технології діаграми Ґанта, використані для вивчення обраних предметів

Діаграма Ґанта — вид стовпчастої діаграми, що застосовують для зображення планів проектів та робочих планів.

Діаграма Ґанта — інтсрумент для планування та управління проектів. Перша Діаграма Ґанта була створена у 1910 році Генрі Ґантом. Тоді їх створювали на папері. Після появи комп’ютерів, приблизно у 1985 році, діаграма Ґанта удосконалилася та ставала більш деталізованою. Сьогодні ними користуються більше за будь-який інший інструмент для планування проекту.

Діаграма Ґанта створюється за допомогою відрізка, який розміщують поруч із горизонтальною шкалою часу. Кожен відрізок має окреме завдання або підзадачу. Найчастіше їх створюють за допомогою таких правил: зліва знаходиться

список завдань, справа розташована шкала часу, на якій підписана робота. Завдання, складова плану та підзадача розміщуються вертикально. Шкала часу має такі параметри: початок, завершення та час завершення задачі. Деякі діаграми Ґанта зображують точки завершення та зв’язки між завданнями [3].

Приклад діаграми Ґанта зображено на рисунку 1.3.

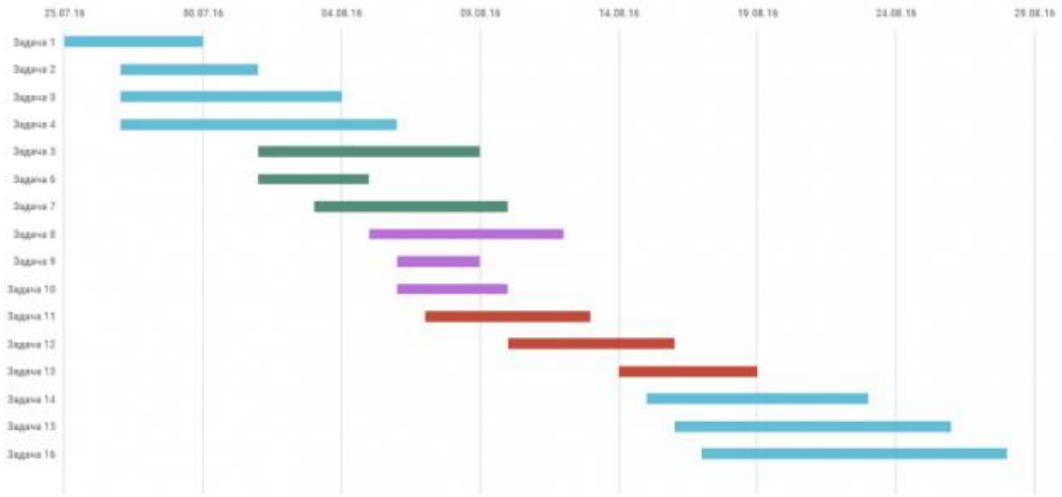

Рисунок 1.3 — Діаграма Ґанта

Важливий елемент діаграми Ґанта — віха, тобто маркер, що позначає важливий момент у робочому процесі, а також спільні межі для завдань. Віха використовується, для візуалізації зв’язків при виконанні завдань. Віха або інші межі діаграми Ґанта, не буває датою календаря. Якщо віха буде неправильною, тоді весь проект буде порушеним, тому діаграму Ґанта не застосовують для плану роботи. Крім того, діаграми Ґанта не вирішують проблеми із ресурсоємністю та важливістю характеру роботи. Під час аналізу детального проекту, діаграма Ґанта є дуже складною, тому втрачає актуальність.

Діаграмою Ґанта найбільше користуються у сфері, пов’язаній з проектами. За допомогою неї, є можливість організувати важкі плани, що складаються з двох або більше команд та можуть змінити термін вивчення. Діаграма Ґанта виконує роботу планувальника часу, яка здатна зберігати усі зазначені терміни та розумно користуватися ресурсами.

Діаграма Ґанта є дуже універсальною. Проте основною проблемою характеризують те, як чітко необхідно вказати завданя та процес виконання. За допомогою них можна оцінити проект та обсяг термінів завершення проекту. Візуалізація буде гіршою, якщо кількість задач буде великою.

Іноді виникають складності із термінами виконання задачі, проте, було створено прості та доступні інструменти управління проектами, що використовують діаграму Ґанта для візуального відображення завдань.

Діаграма Ґанта відстежує та показує:

— завдання, що створені для проекту;

— час початку та завершення завдання;

— дата початку та завершення проекту;

— залежності між конкретними задачами;

— час виконання кожної задачі.

Перевагою діаграми Ґанта є графічне зображення та широкий спектр користування. Графічне зображення достатньо зрозуміле користувачу. Діаграма створена за допомогою горизонтальних смуг, які позначають різними кольорами, де створена смуга належить роботі конкретного користувача або відповідає за певний вид діяльності. Широкий спектр використовується, коли діаграму застосовують для планування завдань, завдяки чому графік стає загальним вирішенням для поточного завдання.

1.3 Огляд математичних моделей визначення пріоритетів при виконанні різного роду завдань

Під час вирішення проблем або поставлених завдань користувачів виникають запитання про порядок реалізації: у якій послідовності планувати вирішення поставлених проблем та які з них відкладаються на більш тривалий час. Так як вирішити всі помилки одночасно неможливо, потрібно проаналізувати, яку із помилок необхідно виправити в першу чергу, для того щоб вирішити найбільш серйозні проблеми.

Важливість завдання залежить від того, чи буде досягнута поставлена мета. Обов’язкове завдання, у якому неможливо досягнути мети не вирішивши поставлену задачу, має високий рівень важливості. Завдання, яке може бути вирішеним без будь-яких дій має середній рівень важливості. Завдання, яке можна зовсім не виконувати для досягнення поставленої мети має низький рівень важливості.

Терміновість допомагає визначити, чи необхідно вирішити поставлене завдання та реалізувати його за короткий проміжок часу. Теоретично це звучить так: якщо часу на вирішення поставленого завдання витрачається мало, тоді актуальність такого завдання є високим [4].

Під час виконання поставленої задачі аналізуються два різних типи пріоритетів: статичний та динамічний. Статичне планування застосовують, якщо є вся інформація стосовно завдання, яке необхідно вирішити, а саме час, за який його необхідно вирішити, якою є структура завдання, коли ви знаєте характеристики часу завдання та способи обміну доступною інформацією. Під час створення системи може бути сконструйована статична діаграма, що буде містити загальну інформацію про цикл виконання, тоді планування періоду створення стає простим інтерпретатором статичної діаграми.

Для простих системних алгоритмів, у яких ефективність не є критичною, система пріоритетів не застосовується. Проте більшість алгоритмів використовують саме алгоритм планування на основі пріоритету завдання. Завершені завдання відсортовуються по їхньому пріоритету, тому завдання з найвищим пріоритетом буде найвищим. При розстановці пріоритетів завдань, велике значення мають характеристики часу. Як і планування, пріоритет буває статичним та динамічним. Отже, за способом встановлення пріоритетів поставленим завданням, алгоритми планування поділяються на алгоритми статичного та динамічного пріоритетного планування. Пріоритети для поставлених задач вказують при їх включенні в систему під час розробки, де вони залишаються незмінними протягом усього часу роботи системи.

Алгоритми динамічного пріоритетного планування більш ефективні, але складні у реалізації. Алгоритми статичного пріоритетного планування прості у реалізації, проте є менш ефективними.

У складних системах реального часу, виконання будь-якої критичної задачі повинно гарантувати виконання абсолютно всіх задач роботи підсистеми. Гарантії надаються в результаті:

— вичерпної перевірки всіх задач поведінки об’єкту та процедури контролю;

— складання статичного розкладу;

— обирання алгоритму динамічного планування, який є математично обґрунтованим.

Під час вибору алгоритму планування необхідно врахувати будь-які можливі зв’язки задач. Задача є залежною, коли створюється залежність: критичний розділ або обмеження порядку виконання.

Отже, під час планування залежних між собою завдань виникає серйозна проблема, яка вирішується під час:

— поділу проблеми планування на дві окремі частини, у якому одна частина проблеми є достроково виконаною;

— обмеження поведінки набору завдань.

Якщо дотримуватись вищезазначених правил, тоді планування буде наближене до статичного.

Якщо терміни часу дотримані, тоді розклад є правильним. Набір завдань $[T_n]$ є допустимим розкладом, якщо певний алгоритм планування дотримується коректного розкладу обраного набору завдань.

Розглянемо систему окремих задач періодичності $[T_n]$. Проте, цей набір завдань може мати непередбачувані випадки, наприклад спорадичність та аперіодичність задачі.

Розклад задач, які повинні повторюватись, представляє собою створену таблицю, яка вказуватиме, яке завдання має виконуватись на даний момент. Гіперперіодом такого розкладу є час створення розкладу, що буде найменшим

спільним кратним періодом завдань, які потребують розв'язку. Отже, таке планування повторює послідовність завдань, зазначених у створеному розкладі.

Розглянемо приклад статичного розкладу, що складається із чотирьох завдань: $T_1[*, 5, *, 2]$; $T_2[*, 6, *, 2.8]$; $T_3[*, 26, *, 2]$; $T_4[*, 26, *, 3]$.

Переваги такого алгоритму:

— просте передавання управління завданнями;

— результати випробувань та планування будуть перевіреними.

Такий тип алгоритму використовують коли необхідна висока надійність.

Недоліки такого алгоритму:

— відсутність гнучкості;

— планування є відокремленим тому що воно виконується коли відбувається переривання часу;

— розклад таблиці можливо буде великим за розміром.

Розглянутий тип планування обробляє переривання часу, тобто завдання є лише підпрограмою, яка виконується обробником переривання у відповідний час виконання.

Така система завдань має гіперперіод — 26. У такому випадку система поставленої задачі отримає контроль 6 разів, друга задача отримає контроль 5 разів, третя та четверта задача — по два рази. Усі терміни виконання поставлених задач повинні бути збереженими та дотриманими. Розклад створеної системи завдань наведено у таблиці 1.1, де позначення «І» означає неактивну задачу.

Оскільки таймери спрограмовано на генерацію переривань не через певний проміжок часу від старту, а через проміжок часу заданого конкретного моменту, практично наведену таблицю краще використовувати по-іншому: необхідно замінити проміжки таймеру початкового циклу на відносні часові інтервали одного переривання до іншого. Оновлений приклад створеної системи завдань наведено у таблиці 1.2.

Таблиця 1.1 — Розкладу створеної системи завдань

№	Час	Задача	№	Час	Задача
1	0	T_1	10	10.8	I
2	1.2	T_3	11	12	T_2
3	2.4	T_2	12	13.2	T_1
4	3.6	I	13	14.4	I
5	4.8	T_1	14	15.6	T_1
6	6	I	15	16.8	I
7	7.2	T_4	16	18	T_2
8	8.4	T_2	17	19.2	I
9	9.6	T_1			

Таблиця 1.2 — Оновлений приклад створеної системи завдань

№	Час	Задача	№	Час	Задача
1	0.4	T_1	10	1.2	I
2	1.2	T_3	11	1.4	T_2
3	1.2	T_2	12	2	T_1
4	1.6	I	13	1.2	I
5	0.4	T_1	14	1.4	T_1
6	1.2	I	15	1.2	I
7	1.2	T_4	16	1.2	T_2
8	2.2	T_2	17	2	I
9	1.6	T_1			

Якщо було змінено кількість задач або терміни виконання завдання, тоді необхідно виконати перепланування, що дуже ускладнить виконання спорадичних завдань.

Динамічне планування динамічних пріоритетів. Було створено два види динамічного планування динамічних пріоритетів:

— EDF (earliest deadline first);

— LLF (least laxity first).

При плануванні видом EDF пріоритетність задач встановлюється, коли у будь-який час найвищий пріоритет має завдання, яка потребує найменші терміни виконання. Видозміни існують зі зміщенням задач та без нього. При плануванні видом LLF пріоритетність задач встановлюється, коли в будь-який час найвищий пріоритет має завдання з найменшим залишком часу.

Доведення: якщо можна сформулювати коректний розклад для набору завдань, тоді він буде приведений до такої форми, що порядок завдань залишиться таким же, як у плануванні видом EDF [5].

На рисунку 1.4 зображено діаграму резерву часу, яка підтверджує концепцію.

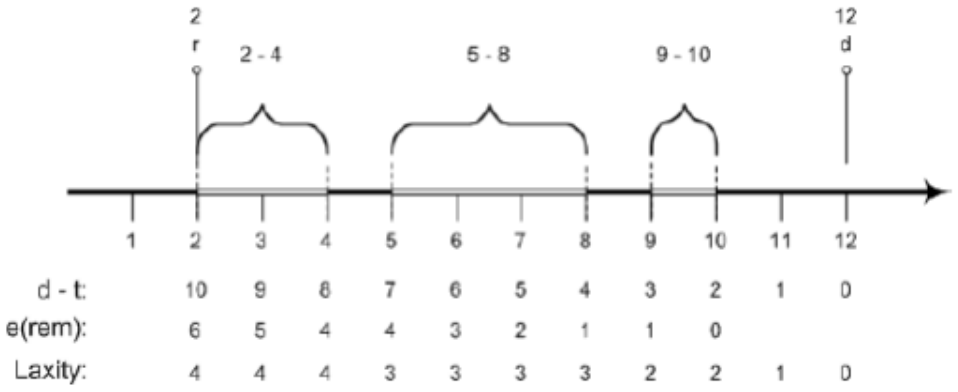

Рисунок 1.4 — Діаграма резерву часу

Діаграма показує роботу над задачею, яку було завершено у момент часу 2 із кінцевим терміном 12, отже завдання завершило переміщення. Під час виконання роботи, терміни за час незмінюються, тому що час завершення та загальний термін часу, що залишилось опрацювати зменшується на однакову суму. Якщо ж задача простоює через витіснення її іншими, більш пріоритетними завданнями, то крайній термін наближається, а час, який ще потрібно опрацювати, залишається постійним, тому в такі проміжки запас часу зменшується. Аналогічна теорема застосовується

до алгоритму планування видом LLF.

Резерв часу — різниця між часом, що залишається до кінцевого терміну та часом у якому завдання ще повинне завершитись (наведено у формулі 1.1):

$$L(t) = (d - t) - e\ (rem) \tag{1.1}$$

Алгоритм планування виду EDF без зміщення призводить до неправильного планування. Доведемо твердження, тому наведемо контрприклад, шляхом постановки набору завдань, яким можна створити правильний розклад, в той час як алгоритм планування виду EDF без зміщення призводить до неправильного планування.

Розглянемо приклад розкладу, у який занесено три завдання, де параметрами є: $T_1: r, e, d = 0, 3, 10;\ T_2: r, e, d = 2, 6, 14;\ T_3: r, e, d = 4, 4, 12.$

Приклад розкладу створеного для такого набору завдань зображено на рисунку 1.5.

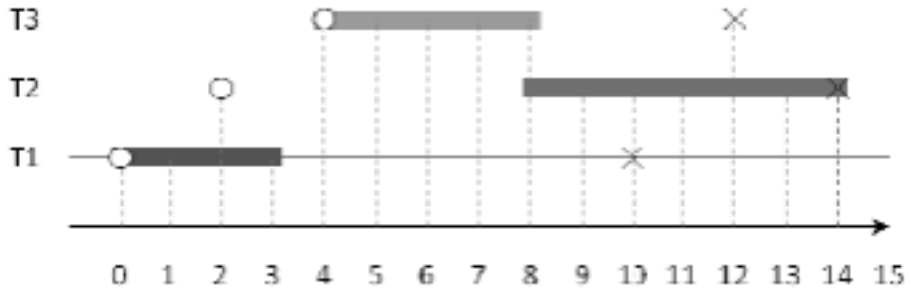

Рисунок 1.5 — Приклад розкладу для набору, у який занесено три завдання

Приклад розкладу, який було створено за допомогою алгоритму планування виду EDF без зміщення зображено на рисунку 1.6.

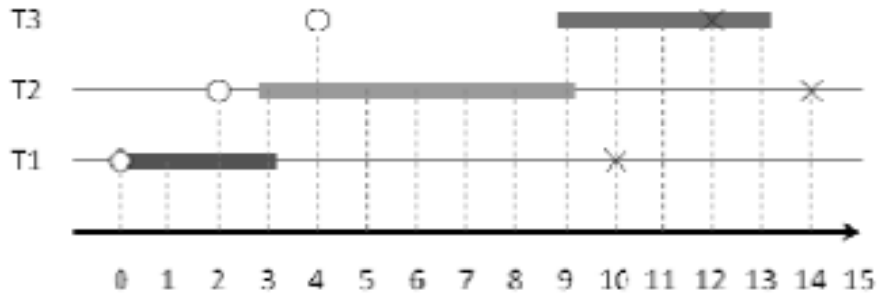

Рисунок 1.6 — Приклад розкладу, який було створено за допомогою алгоритму планування виду EDF без зміщення

У момент часу 3 виконано тільки завдання 2, отже його обирають для завершення , оскільки завдання не зміщується, поки не виконає свою задачу повністю, навіть тоді, коли у момент часу 4 завдання 3 виконується у стані готовності з терміном виконання меншим, за терміни завдання 2. Отже, завдання 3 не встигає завершити виконання поставленої задачі. Тому, алгоритм планування виду EDF без зміщення не є актуальним.

Приклад розкладу, який було створено за допомогою алгоритму планування виду EDF із можливістю зміщенням завдань зображено на рисунку 1.7.

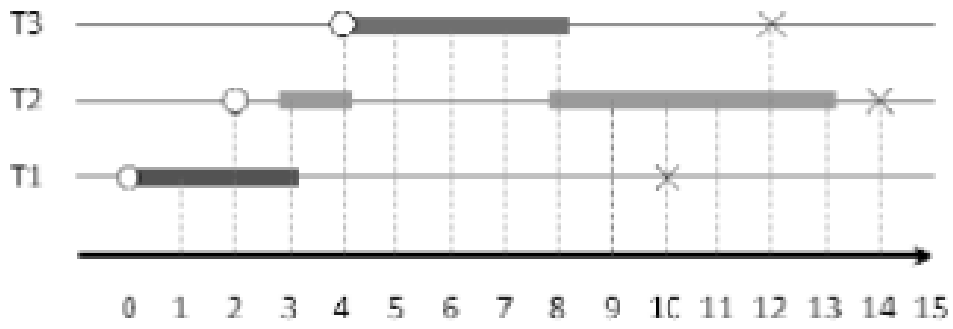

Рисунок 1.7 — Приклад розкладу, який було створено за допомогою алгоритму планування виду EDF із можливістю зміщенням завдань

За допомогою алгоритму планування виду EDF із можливістю зміщенням

завдань, поставлені задачі виконуються раніше критичного терміну. У момент часу 2 було завершене завдання 2, яке після цього змішує завдання 1, тому що час виконання завдання 1 настає раніше за термін завдання 2. Тому, завдання 2 контролюється лише в момент часу 3. У момент часу 4 створюється завдання 3, яке зміщує завдання 2, тому що завдання 2 має пізніший крайній термін задачі, ніж у завдання 3. У момент часу 8, завершує роботу завдання 3, тому повний контроль передається завданню 2.

Однак алгоритм планування виду EDF без зміщення не є гіршим за алгоритм планування виду EDF із зміщенням, тому що припущення оптимальності алгоритму із зміщенням, базується на думці, що зміщення завдання не потребує витрати додаткового часу, що є неправильним.

Динамічне планування статичних пріоритетів. Було створено два види динамічного планування статичних пріоритетів:

— RMS (rate monotonic scheduling);

— DMS (deadline monotonic scheduling).

Пріоритети алгоритму планування виду RMS визначаються за твердженням: завдання, яке має коротший відносний термін виконання — має вищий пріоритет. Тобто, якщо завдання постійно готове до виконання, тоді пріоритет такого завдання буде вищим. У алгоритмі планування виду DMS навпаки, якщо терміни виконання поставленого завдання є короткими, тільки тоді таке завдання має вищий пріоритет.

Приклад розкладу, який було створено за допогомою алгоритму планування виду RMS для синхронної системи з трьох періодичних задач з такими параметрами: T_1: (3; 0,5); T_2: (4; 1); T_3: (6; 2), зображено на рисунку 1.8.

Кінцевий термін розкладу алгоритму планування виду RMS дорівнює періоду. Відповідно до наведених тверджень, завдання 1 є із найвищим пріоритетом, в той час, як завдання 3 має найнижчий пріоритет. У момент часу 0 усі три завдання є завершеними, тому контроль має завдання 1. У момент часу 0,5 готовими є завдання 2 та завдання 3, в такому випадку контроль має завдання 2. У

момент часу 3 завершується виконання завдання 1, тому воно зміщує завдання 3. У момент часу 3,5 контроль має знову завдання 3, тому шо воно єдине завершене. В момент часу 4 виконане завдання 2, тому воно має контроль. У момент часу 6 завершеними є завдання 1 та завдання 3, тому контроль отримує завдання 1. Потім виконується завдання 3, але в момент часу 8 воно зміщується завданням 2. В момент 9 завершеним є завдання 1, в момент часу 9,5 контроль отримує завдання 3.

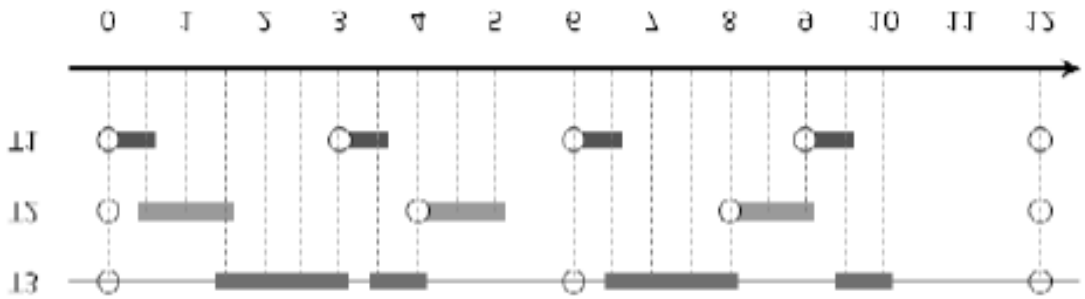

Рисунок 1.8 — Приклад розкладу, який було створено за допомогою алгоритму планування виду RMS

Приклад розкладу, яккий було створено за допомогою алгоритму планування виду DMS із наступними параметрами: T_1 (3; 0,5); T_2 (4; 1); T_3 (6; 2), зображено на рисунку 1.9.

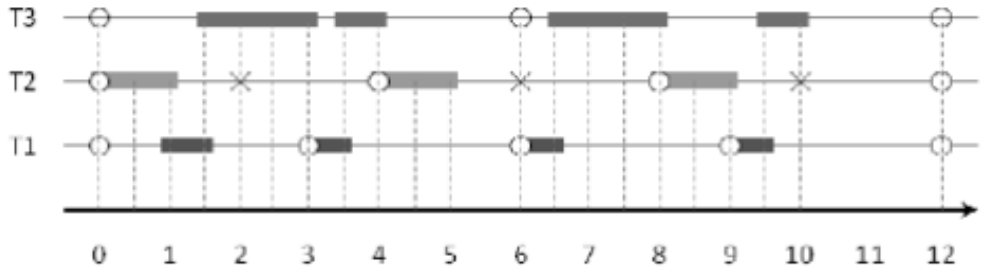

Рисунок 1.9 — Приклад розкладу, який було створено за допомогою алгоритму планування виду DMS

Порівняльна таблиця розглянутих математичних моделей наведена у таблиці 1.3.

Таблиця 1.3 — Порівняльна таблиця математичних моделей планування

Алгоритм планування	**Простота реалізації**	**Ефективність та гнучкість**	**Надійність результатів планування**
Статичне планування	+	-	+
Динамічне EDF-планування	-	+	+
Динамічне LLF-планування	-	+	-
Динамічне RMS-планування	-	+	-
Динамічне DMS-планування	-	+	+

1.4 Порівняльний аналіз аналогів програмного забезпечення, які користуються діаграмою Ґанта

Порівняння програмного забезпечення, які користуються діаграмою Ґанта, наведено у таблиці 1.4.

Таблиця 1.4 — Порівняльна характеристика аналогів із підсистемою, що розробляється

Характеристика	**Microsoft Excel**	**Microsoft Power Point**	**Microsoft Visio**	**LibreOffice**	**Gantt Project**	**TeamGantt**	**Підсистема, що розробляється**
Вказування термінів	+	+	+	+	+	+	+
Шкала виконання	+	+	+	-	+	-	+
Статичне планування	+	+	+	+	+	+	+
Редагування термінів	-	-	-	-	+	-	+
Повний доступ	+	+	+	+	-	-	+

Програмне забезпечення Microsoft Excel здійснює пошуки різних моделей діаграми Ґанта. Для цього необхідно ввести у рядку пошуку слова «діаграма Ґанта». Після чого оберіть запропоновану модель діаграми, які видало вам програмне забезпечення. Більша частина роботи буде виконана за вас, тож необхідно буде лише налаштувати клітинки, які містять назву проекту. Відредагуйте усі назви заходів, налаштуйте початкову дату та заплановану тривалість. Створіть дату завершення та частку завершення налаштованої діяльності. Усі зміни у клітинках відповідатимуть за зміни, які будуть зображені праворуч, тобто створеною діаграмою Ґанта.

Програмне забезпечення Microsoft PowerPoint дозволяє відредагувати назву та будь-яку іншу дію після відкриття діаграми Ґанта. Проаналізуйте назви усіх можливих дій, вказаний час рядків діаграми та перевірте діаграму на наявність усіх внесених дій користувача, які будуть виділеними жовтим ромбом. Перейдіть на наступний слайд, там ви побачите легенду діаграми, де ви зможете зрозуміти усі параметри, які було внесено у діаграму.

Програмне забезпечення Microsoft Visio пропонує ввести дані діаграми, одразу після її створення. Діаграма складається з таких параметрів, як обсяги діяльності, дата створення проекту, дата завершення проекту та прогнозовані терміни виконання поставленої задачі. Після завершення початкової компіляції, натисніть «ОК», для того щоб створити діаграму Ґанта. Якщо вам потрібно внести деякі зміни у діаграму, для цього натисніть на точку створеної діаграми або використайте бокову панель, яка складається із кнопок керування обраного типу діаграми.

Програмне забезпечення LibreOffice дозволяє редагувати будь-яку комірку, що можна заповнити параметрами. Встановлюйте назву графіка, час проекту та ефективну тривалість. Для того, щоб змінити створену діаграму скористайтеся таблицею, яка розташована у правій частині програмного забезпечення, яке створене для тих, кому необхідно автоматично створювати нескладні та зрозумілі діаграми Ґанта.

Програмне забезпечення GanttProject дозволяє виконати планування великих проектів, за допомгою поділу на декілька підзадач. Переглядайте діаграми та взаємозв'язки між різними датами календаря, видами завдань та споживчими ресурсами. У створеній діаграмі Ґанта є можливість аналізу часу кожного виду завдань. Контролюйте та відстежуйте їх зміну в часі. Створені проекти можуть налаштовуватися детально. Будь-які внесені зміни, що були занесені до проекту, можуть бути збережені [6].

Програмне забезпечення TeamGantt використовується для керування проектами одного або двох користувачів. Оскільки програмне забезпечення не є безкоштовним, у ньому доступні передплати для того щоб використовувати програмне забезпечення без обмежень. Після запуску TeamGantt відкривається інтерфейс, схожий на інтерфейс програмного забезпечення Microsoft Project. Починайте створювати діаграму Ґанта для керування вашого проекту. Навіть для планування та управління важких проектів та задач використовувати програмне забезпечення TeamGantt дуже просто. Після чого створіть список заходів, упорядкуйте усі ресурси, які використовуються та розділіть ваш проект на суб-заходи, для цього встановивши власні терміни та умови.

2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ ПІДСИСТЕМИ КОРИГУВАННЯ ТЕРМІНІВ ВИВЧЕННЯ ОБРАНИХ ДИСЦИПЛІН СИСТЕМИ ДИСТАНЦІЙНОГО НАВЧАННЯ

2.1 Побудова моделі врахування змін у термінах вивчення дисциплін

Поняття часу відіграє важливу роль у сучасному світі, оскільки його неможливо зупинити або отримати від нього матеріальну вигоду, як від грошей. Тому залишається користуватись ним незмінною ціною, тобто рівно шістдесят секунд на одну хвилину. Час неможливо включити, неможливо виключити, і найголовніше — неможливо замінити. Час можна охарактеризувати негнучким елементом людського існування. Єдине шо з ним можна робити, так це проаналізувати як ми його використовуємо, тобто ефективним він є чи ні.

Тайм-менеджмент — процес навчання або практика, яка допомагає використовувати час, витраченого на конкретну діяльність, завдяки чому підвишується продуктивність та ефективність.

Складання розкладу дозволяє створювати власні терміни навчання, зосереджуючись на темах, які представляють власні інтереси. Оптимізований розклад скорочує щоденні витрати часу на різні предмети або теми кожного дня і допомагає виділити час на повторення обраних тем створеної дисципліни. Не потрібно відвовити відведений час на теми предметів, що не будуть вивчатися на професійному рівні згідно із навчальним планом.

Після створення розкладу для вивчення обраних дисциплін, потрібно уважно розподілити увесь час на вивчення кожного предмету. Менше часу слід приділяти вступним та заключним заняттям обраного предмету для вивчення, краще приділіть більше часу на вивчення основних та важливих для розуміння тем, які можуть найкраще допомогти для розуміння вивченого матеріалу будь-якого предмету для вивчення [7].

Щоб підсистема була ефективною, модель врахування змін у термінах вивчення дисциплін будується з урахуванням таких умов:

Студент не повинен витрачати більше часу за вказаний на вивчення дисциплін:

$$T_1[s][t] + T_2[s][t] + T_3[s][t] + T_n[s][t] + \ ... < T_d[\text{в}], \qquad (2.1)$$

де T_1, T_2, T_3, T_n — час, який студент вказує на вивчення обраної теми одного предмету для вивчення. Кількість тем предметів для вивчення залежить від значення максимального часу, який студент може витратити на навчання для конкретного дня тижня;

s — індекс обраного предмету для вивчення;

t — номер теми обраного предмету для вивчення;

$T[\text{в}]$ — максимальний час, який студент може витрати за один день на вивчення обраних дисциплін;

d — день тижня.

Максимальний час, який студент може витрати на вивчення дисциплін в конкретний день тижня розраховується за формулою:

$$T_d[\text{в}] = T[\text{дн}] * x_d[\text{вк}], \qquad (2.2)$$

де $T[\text{дн}]$ — час повного дня, який дорівнює 24 години, або 1440 хвилин;

$x_d[\text{вк}]$ — коефіцієнт часу, який залежить від того, скільки часу готовий витрати студент на вивчення в конкретний день тижня.

За замовчуванням $x_d[\text{вк}] = 1$, тобто 24 години. Наприклад, якщо студент готовий витрати на вивчення дисциплін в обраний день тижня 6 годин, тоді $x_d[\text{вк}]$ буде дорівнювати 0,25. Наступного дня, студент може вказати інше значення, тому щодня максимальний час, який студент може витрати на вивчення дисциплін буде

різним.

Очевидно, що $x_d[\text{вк}]$ не може перевищувати значення 1, оскільки повний день має не більше, ніж 24 години, тому $x_d[\text{вк}] < 1$.

Час вивчення наступної теми обраної дисципліни автоматично змінюватиметься після вивчення попередньої теми обраної дисципліни.

Припустимо, що студент справився швидше з вивченням обраної теми дисципліни, тоді до наступної теми обраного предмету, весь час, що залишився — додається:

$$\mathrm{T}_{n+1}[\mathrm{s}][\mathrm{t}+1] = \mathrm{T}_{\mathrm{n}+1}[\mathrm{s}][(t+1)_{\text{запл}}] + \mathrm{T}_{\mathrm{n}}[s][t_{\text{вільн}}], \qquad (2.3)$$

де $\mathrm{T}_{n+1}[\mathrm{s}][\mathrm{t}+1]$ — час вивчення наступної теми обраної дисципліни;

$\mathrm{T}_{\mathrm{n}+1}[\mathrm{s}][(t+1)_{\text{запл}}]$ — час, який було заплановано студентом для вивчення наступної теми обраної дисципліни;

$\mathrm{T}_n[s][t_{\text{вільн}}]$, — вільний час, який залишився у студента від попередньої теми обраної дисципліни для вивчення.

Вільний час, який залишився у студента від попередньої теми обраної дисципліни для вивчення розраховується за формулою:

$$\mathrm{T}_{\mathrm{n}}[s][t_{\text{вільн}}] = \mathrm{T}_{\mathrm{n}}[s][t_{\text{запл}}] - \mathrm{T}_{\mathrm{n}}[s]\left[t_{\text{реал}}\right], \qquad (2.4)$$

де $\mathrm{T}_n[s][t_{\text{вільн}}]$ — вільний час, який залишився у студента від попередньої теми обраної дисципліни для вивчення.

$\mathrm{T}_{\mathrm{n}}[s][t_{\text{запл}}]$ — запланований час, який студент повинен був витрати на вивчення попередньої теми обраної дисципліни для вивчення.

$\mathrm{T}_{\mathrm{n}}[s]\left[t_{\text{реал}}\right]$ — реальний час, який студент витратив на вивчення попередньої теми обраної дисципліни для вивчення.

Якщо студент витратив більше часу за запланований час для вивчення обраної теми дисципліни, тоді зайвий час, який студент витратив —віднімається від запланованого часу у наступної теми обраної дисципліни:

$$\mathrm{T}_{n+1}[\mathrm{s}][\mathrm{t}+1] = \mathrm{T}_{\mathrm{n}+1}[\mathrm{s}][(t+1)_{\text{запл}}] - \mathrm{T}_{\mathrm{n}}[s][t_{\text{зайв}}], \qquad (2.5)$$

де $\mathrm{T}_{n+1}[\mathrm{s}][\mathrm{t}+1]$ — час вивчення наступної теми обраної дисципліни;

$\mathrm{T}_{\mathrm{n}}[s][(t+1)_{\text{запл}}]$ — запланований час, який студент повинен був витрати на вивчення наступної теми обраної дисципліни для вивчення.

$\mathrm{T}_{\mathrm{n}}[s][t_{\text{зайв}}]$ — зайвий час, який студент витратив на вивчення попередньої теми обраної дисципліни для вивчення.

Зайвий час, який студент витратив на вивчення попередньої теми обраної дисципліни для вивчення розраховується за формулою:

$$\mathrm{T}_{\mathrm{n}}[s][t_{\text{зайв}}] = \mathrm{T}_{\mathrm{n}}[s]\left[t_{\text{реал}}\right] - \mathrm{T}_{\mathrm{n}}[s][t_{\text{запл}}], \qquad (2.6)$$

де $\mathrm{T}_{\mathrm{n}}[s][t_{\text{зайв}}]$ — зайвий час, який студент витратив на вивчення попередньої теми обраної дисципліни для вивчення.

$\mathrm{T}_{\mathrm{n}}[s]\left[t_{\text{реал}}\right]$ — реальний час, який студент витратив на вивчення попередньої теми обраної дисципліни для вивчення.

$\mathrm{T}_{\mathrm{n}}[s][t_{\text{запл}}]$ — запланований час, який студент повинен був витрати на вивчення попередньої теми обраної дисципліни для вивчення.

2.2 Побудова моделі навчання

Головне завдання під час створення розкладу — забезпечення та планування усіх навчальних предметів для методичного вивчення, тобто дотримання усіх зв’язків, коректної послідовності та забезпечення форм обраних навчальних дисциплін.

При організації розкладу вивчення обраних предметів для вивчення, необхідно дотримуватись таких умов забезпечення:

— рівномірно розподіліть навантаження на вивчення навчального предмету;

— створіть гнучкий графік навантаження.

Намагайтеся витрачати менше часу на вивчення дисциплін спочатку, але потім збільшуйте рівномірно час для вивчення головних тем усіх створених дисциплін. Дотримуйтеся балансу, для того щоб не перевантажувати себе під час вивчення навчальних дисциплін [8].

Самостійне створення термінів розкладу для вивчення обраних дисциплін потребує велику кількість часу. Оскільки неможливо розглянути усі обмеження, кінцевий результат може не задовільнити потреби користувача, тому дуже важливо спланувати правильну автоматизацію навчання. Остаточні терміни виконання роботи визначаються саме під час планування, але якщо вивчення обраної дисципліни не може бути завершеним в обумовлені терміни, тоді планування дозволяє відкласти вивчення обраного предмету.

Було створено декілька підходів до розв'язання поставленого завдання планування та створення розкладу. Методи імітації випалювання та алгоритми розфарбовування графу є ефективними, щоб створити декілька простих розкладів. Коли алгоритми реалізуються, на основі принципів імітаційного моделювання, можливість використання розробленої підсистеми стає обмеженим. При мінімальних змінах у компонентах підсистеми, вимагатимуться серйозні зміни у створеному алгоритмі [9].

В моделях навчання, які будуть розглядатись далі максимальною пропускною здатністю є увесь допустимий час для вивчення обраних дисциплін на один день, тобто $T[\text{дн}]$; фактичною пропускною здатністю є максимальний час, який студент може витрати на вивчення дисциплін в конкретний день тижня, тобто $T_d[\text{в}]$; кількістю умовних одиниць на які можна збільшити пропускну здатність є вільний час, який залишився у студента від попередньої теми обраної дисципліни для вивчення, тобто $\mathrm{T_n}[s][t_{\text{віл}}]$ (див. пункт 2.1).

Щоб навчання студента було ефективним, існують різні можливі варіанти вивчення обраних дисциплін системи дистанційного навчання. Розглянемо дві найбільш ефективних моделі навчання.

Студент вивчає всі доступні теми обраного предмету по черзі, і лише потім переходить до вивчення наступного предмету. Приклад графової моделі даного варіанту зображена на рисунку 2.1.

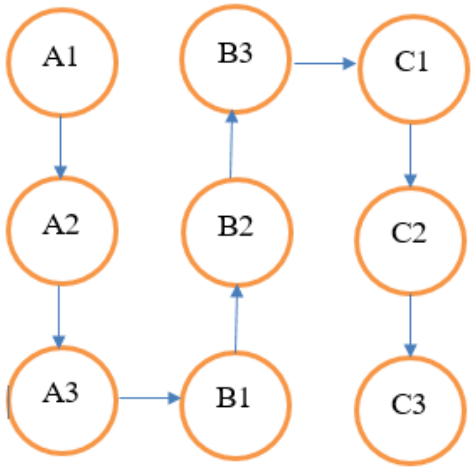

Рисунок 2.1 — Графова модель вивчення обраних предметів по черзі

де A1, A2, A3 — перша, друга та третя теми першого обраного предмету для вивчення;

B1, B2, B3 — перша, друга та третя теми другого (наступного) обраного предмету для вивчення;

C1, C2, C3 — перша, друга та третя теми третього (наступного) обраного предмету для вивчення.

Студент вивчає обрані предмети згідно університетському плану, тобто спочатку він вчить перші теми кожного із обраних предметів для вивчення, і лише

потім переходить до вивчення наступних тем усіх обраних предметів для вивчення. Приклад графової моделі даного варіанту зображена на рисунку 2.2.

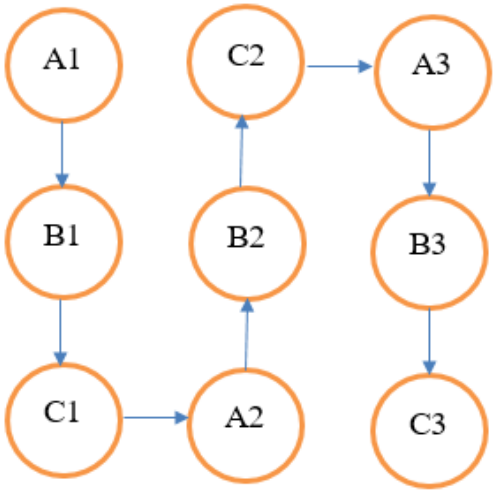

Рисунок 2.2 — Графова модель вивчення обраних тем по черзі

де A1, B1, C1 — перші теми першого, другого та третього обраного предмету для вивчення;

A2, B2, C2 — другі (наступні) теми першого, другого та третього обраного предмету для вивчення;

A3, B3, C3 — треті (наступні) теми першого, другого та третього обраного предмету для вивчення.

Зауважимо, що і у першої, і у другої графових моделей можливо проходити через вершину графа лише один раз.

2.3 Реалізація алгоритму роботи підсистеми

Створення мобільної підсистеми — галузь програмування, яка швидко розвивається, тому що частка мобільних телефонів є більшою за частку користувачів ноутбуків та персональних комп’ютерів, так що актуальність напрямку, що розглядається стає більшою кожного дня.

Завдання розробника мобільної підсистеми — створити придатну для користування підсистему на мобільну платформу Android, що буде багатофункціональною та зрозумілою для користувачів створеної підсистеми.

Процес розробки підсистеми, що розробляється, відбувається з таких етапів:

— складання технічного завдання підсистеми;

— складання процесу та стадій функціонування підсистеми;

— будування архітектури підсистеми;

— створення алгоритму роботи підсистеми;

— програмування підсистеми;

— підтримка та оновлення підсистеми;

— налагодження та тестування підсистеми;

— створення інструкції щодо використання готових підсистем для Android;

— оформлення документаці підсистеми;

— завантаження підсистеми в Google Play Market.

Головна функція розробки підсистем для мобільних пристроїв – розмір екрану, для якого створюється підсистема. Під час створення підсистеми потрібно мінімізувати кількість натиснення кнопок, проаналізувати які потрібно створити та з’єднати функції у підсистемі, чи є пристрій користувача підсистеми обмеженим розширенням екрану. Розробники мобільних підсистем не обмежуються лише програмуванням підсистем, тому що створення підсистем для мобільних пристроїв стосується також креативу та охоплює різні види діяльності.

Важливою функцією для створення підсистеми є під’єднання до мережі Інтернет. Спочатку мобільні пристрої функціонували дуже повільно, але якщо використовувати сучасні багатоядерні процесори і велику кількість оперативної пам'яті тоді розробка підсистеми стане ефективною. Більша частка обчислювальної логіки у підсистемі, що розробляється, стосується саме зв’язку між мережею Інтернет та хмарною службою [10].

Розробнику підсистеми мобільних пристроїв необхідно володіти технічними

навичками, для того щоб бути успішним у роботі в команді. Розробник підсистеми повинен:

— володіти логічним мисленням;

— бути досвідченим у комунікації;

— бути креативним та творчим;

— хотіти та вміти самовдоконалюватися;

— бути уважним та відповідальним.

Розробнику підсистеми необхідно володіти англійською мовою на середньому рівні. Платформи мобільних пристроїв дуже швидко прогресують, оскільки щороку створюються нові версії Android. Інструкція для нових версій знаходиться у мережі Інтернет, яка є написаною англійською мовою. Отже, розробник повинен володіти англійською мовою для того, щоб вивчати документацію, використовувати нові інструменти та функції підсистеми, оновлювати компоненти під час розробки підсистеми.

По-справжньому якісна підсистема — не лише підсистема, що функціонує безпомилково, це якісний інтерфейс підсистеми та дотримання логіки, яка зможе прорахувати будь-який результат роботи підсистеми [11].

Необхідно використовувати формальну модель, щоб виражати необхідні аспекти розробки підсистеми, для того щоб знайти розв’язок та побудувати алгоритм моделі, що розробляється. Побудувавши модель початкового завдання, необіхдно дійти до розв’язку використавши функції моделі, що розробляється.

Головна мета — побудувати розв’язок створивши алгоритм моделі. Розв’язок будується користуючись інструкціями, із чітким значенням, де функціонування відбувається за короткий час, використовуючи обмежені обчислювальні витрати. Вони виконуються безперервно у алгоритмі моделі, де самі визначають, яку кількість разів необхідно виконати повторення.

Незалежно від того, які вхідні дані вводилися, алгоритм моделі буде завершуватися, після виконання необхідної кількості інструкцій. Отже, створений алгоритм не повинен закінчуватися нескінченним циклом обчислень.

На рисунку 2.3 зображено структуру підсистеми коригування термінів вивчення обраних дисциплін.

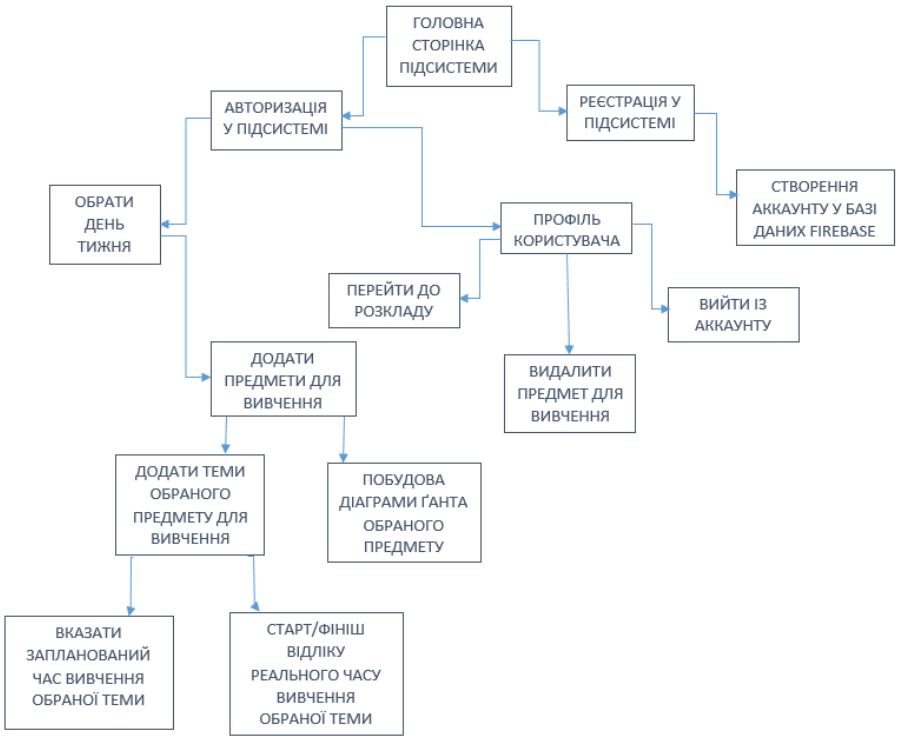

Рисунок 2.3 — Структура підсистеми коригування термінів вивчення обраних дисциплін

Алгоритм можливо визначити іншим способом. Для того, щоб алгоритм був чітким та обчислювався з обмеженою кількістю витрат. Користувач може розуміти деяку частину алгоритму, оскільки для нього вона буде чіткою. Для іншого користувача така же чітка частина алгоритму може бути незрозумілою. Стосовно обмеженої кількості витрат, якщо користувач чітко розуміє, що означає кожна інструкція, йому важко буде пояснити, що інструкція буде виконаною із будь-якими кінцевими даними [12].

Алгоритм — процес, який не тільки гарантує знайде рішення, а ще й гарантує, що знайдене рішення буде оптимальним.

Властивості правильно комбінованого алгоритму:

— обмеження часу;

— правильність;

— визначеныcть;

— скінченість;

— чіткість.

Підсистема працює за наступним алгоритмом. Після запуску підсистеми користувач потрапляє на головну сторінку, підсистема буде чекати дій користувача. Він повинен буде виконати реєстрацію або увійти у створену підсистему. Після успішної реєстрації у підсистемі, дані користувача автоматично потрапляють у базу даних Firebase, використовуючи збережені дані, користувач входить у створену підсистему та починає нею користуватися. Після авторизації користувач переходить до наступних вікон підсистеми, де йому спочатку необхідно обрати день тижня, потім додати один або декілька предметів для вивчення, після чого користувач у створеному предметі користувач створює одну або декілька тем для вивчення, у яких вказує запланований час вивчення обраної теми та розпочинає її вивчення, після натиснення кнопки «Старт». Терміни які ввів користувач зберігаються у базі даних Firebase. Підсистема розпочне зворотний відлік вивчення, та буде зупиненою коли користувач зайде у підсистему знову, натисне на тему, яку вивчає, та виконає натиснення кнопки «Фініш».

За умови, шо користувач вивчив тему швидше, ніж було заплановано, тоді час, який залишився, буде відноситись вже до наступної обраної теми для вивчення. І навпаки, якщо користувач не справився вчасно, тоді лишній час вивчення, буде віднято від наступної обраної теми для вивчення.

Після завершення вивчення обраних тем створеного предмету, підсистема починає будувати діаграму Ґанта, що зображує час вивчення кожної обраної теми створеного предмету зеленим кольором якщо студент справився раніше за заплановані терміни та червоним якщо часу було витрачено більше аніж було заплановано, або діаграма Ґанта не побудується повністю, якщо всі теми не були вивчені студентом. Після побудови діаграми Ґанта, підсистема завершує роботу та звільняє усі ресурси.

Алгоритм керуючої програми коригування термінів вивчення обраних

дисциплін представлено у вигляді UML-діаграми діяльності на рисунку 2.4.

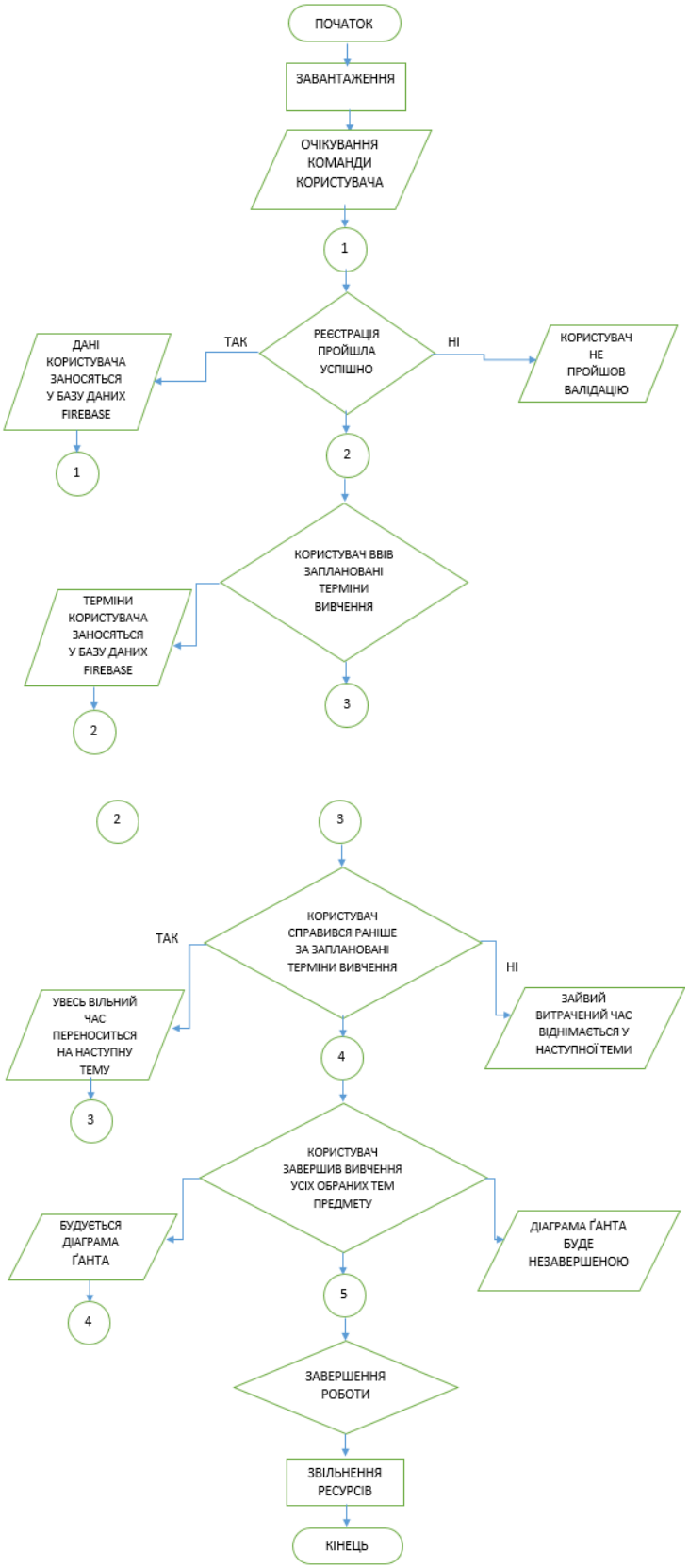

Рисунок 2.4 — UML-діаграма діяльності комп’ютерної програми

2.4 Структура класів підсистеми

UML-діаграма класів програми коригування термінів вивчення обраних дисциплін складається з наступних класів:

— клас головної сторінки програми коригування термінів вивчення обраних дисциплін;

— клас взаємодії із базою даних Firebase;

— клас взаємодії із обраним днем вивчення студента;

— клас взаємодії із обраним предметом вивчення студента;

— клас взаємодії із обраною темою предмету студента;

— клас створення діаграми Ґанта вивченої теми студента.

Опис класів комп’ютерної програми:

MainActivity. Метод onCreate() викликається під час створення чи перезапуску активності. Bundle – клас, який реалізує асоціативний масив, тобто, зберігає пари ключ-значення. Щоб зафіксувати стан активності перед її знищенням, у класі активності необхідно реалізувати метод onSaveinstancestate().

Метод onClick() обробляє натискання на створену кнопку. Метод showSignInWindow(), показує спливаюче вікно з авторизацією користувача. Метод onSuccess() викликається, коли завдання успішно завершується. В якості параметру передається авторизація користувача. Метод onFailure() викликається, коли завдання завершується з помилкою. В якості параметру передається виняток, що поле вводу не є порожнім. Метод showRegisterWindow(), показує спливаюче вікно з реєстрацією користувача. Метод визначення інтерфейсу для зворотного виклику setOnClickListener(), викликається, при натисканні на створену кнопку. В якості параметру створює нову сторінку комп’ютерної програми.

DatabaseTermsActivity. DatabaseTerms() – метод, що взаємодіє із базою даних Firebase. В якості параметрів приймає значення ідентифікаторів, днів, годин та хвилин. Метод getId() дозволяє отримати значення необхідного ідентифікатору із бази даних Firebase. Метод setId() дозволяє записати значення для необхідного

ідентифікатора у базу даних Firebase. Метод getDays() дозволяє отримати значення дня із бази даних Firebase. Метод setDays() дозволяє записати значення дня у базу даних Firebase. Метод getHours() дозволяє отримати значення години із бази даних Firebase. Метод setHours() дозволяє записати значення години у базу даних Firebase. Метод getMinutes() дозволяє отримати значення хвилини із бази даних Firebase. Метод setMinutes() дозволяє записати значення хвилини у базу даних Firebase.

DaysActivity. Метод showProfileWindow(), показує спливаюче вікно профілю користувача. До методу setContentView() передайте ресурс розмітки графічного інтерфейсу.

SubjectActivity. Метод startActivity(Intent) використовується для початку нової дії, яка буде розміщена у верхній частині стека активності. Він приймає один аргумент newIntent(), який описує створення нової дії, яку потрібно виконати. За допомогою методу finish() можна завершити роботу активності.

SubjectTopicActivity. Метод init() — метод ініціалізації об’єкта, який є вже визначеним, коли він створюється. Метод життєвого циклу сервлета Java. Запускається за допомогою браузерного вікна тоді, коли Java виконує завантаження та обробляється браузером. Метод textViewIsNotEmpty() перевіряє чи вписані дані користувачем не є порожніми. Метод onClickDaysChange() обробляє натискання на кнопку із значенням кількості днів, що залишилися до вивчення обраної теми предмету. Метод onClickHoursChange() обробляє натискання на кнопку із значенням кількості годин, що залишилися до вивчення обраної теми предмету. Метод onClickMinutesChange() обробляє натискання на кнопку із значенням кількості хвилин, що залишилися до вивчення обраної теми предмету. Метод onClickSave() обробляє натискання на кнопку із підтвердженням запланованого часу вивчення обраної теми предмету користувачем у будь-яке із попередніх полів.

SubjectGanttDiagramActivity. Метод plantimetopics() показує запланований час вивчення обраного предмету користувачем за допомогою діаграми Ґанта. Метод realtimetopics() показує реальний час вивчення обраного предмету користувачем за

допомогою діаграми Ґанта. Метод changeplantime() автоматично змінює час вивчення наступної теми обраного предмету користувачем, в залежності від успішності із попередньою вивченою темою. Нові значення зображуються за допомогою діаграми Ґанта.

На рисунку 2.5 зображено UML-діаграму класів програми коригування термінів вивчення обраних дисциплін.

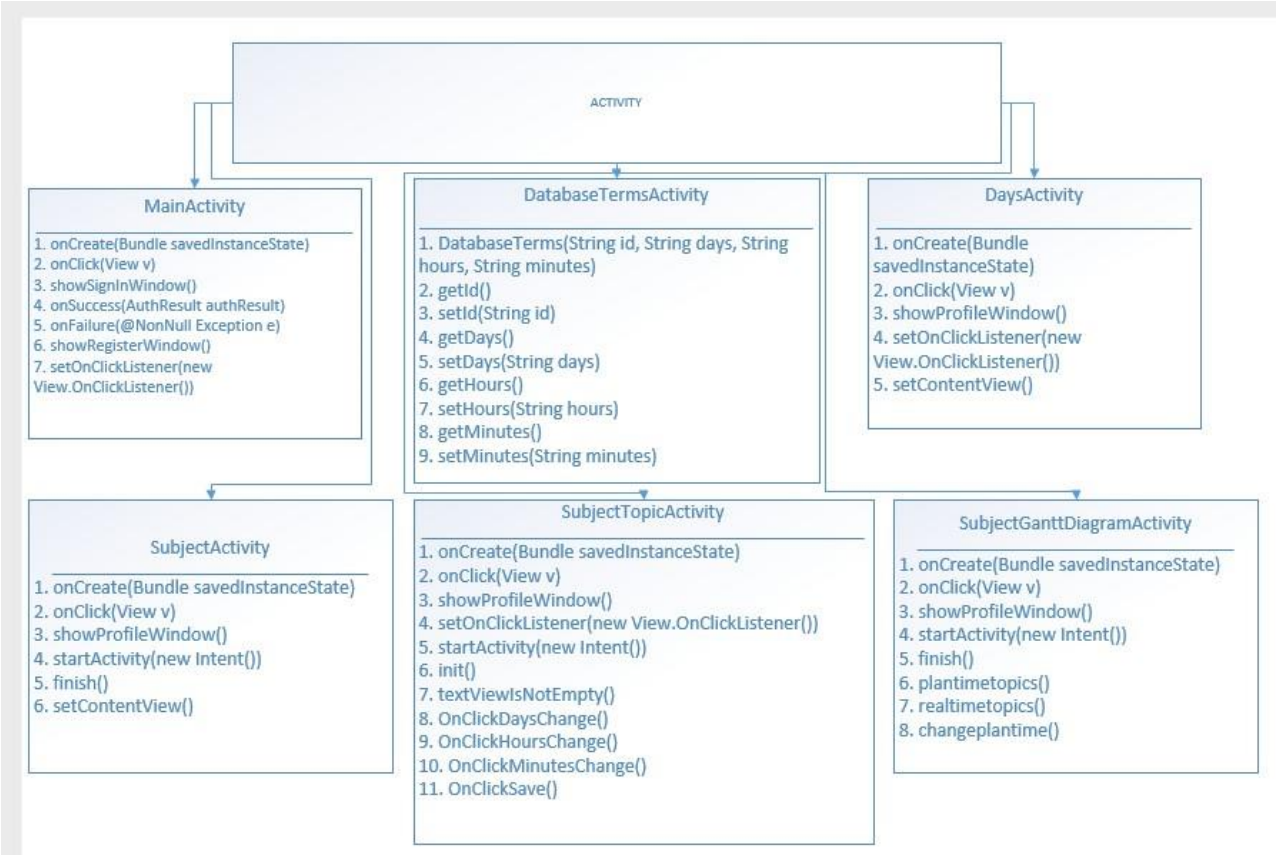

Рисунок 2.5 — UML-діаграма класів програми коригування термінів вивчення обраних дисциплін

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ПІДСИСТЕМИ КОРИГУВАННЯ ТЕРМІНІВ ВИВЧЕННЯ ОБРАНИХ ДИСЦИПЛІН СИСТЕМИ ДИСТАНЦІЙНОГО НАВЧАННЯ

3.1 Обґрунтування вибору програмного середовища та мов програмування

Android Studio — інтегроване середовище розробки, розроблений Google, який допомагає користувачу створювати мобільні підсистеми на Android. Інтегроване середовище розробки базується на програмному забезпеченні IntelliJ IDEA від JetBrains. Android Studio доступне для Windows, Linux і macOS. Android Studio працює на мовах програмування C++, Java та Kotlin.

Середовище розробки встановлюється на персональний комп'ютер безкоштовно. Android Studio має модель, яка створює інтерфейс користувача, з якого робота над підсистемою починається. Середовище надає інструменти створення будь-яких підсистем не тільки для телефонів або планшетів, але й для приладів, які створені на Wear OS, автомобільної системи Android Auto, Google Glass та навіть для телевізорів, які підтримують функцію Android TV. Для підсистем, розроблених за допомогою плагіна ADT Plugin і плагіна Eclipse, з'являється можливість одразу виконати імпорт наявного проекту в середовище розробки Android Studio.

Характеристика IDE Android Studio:

— наявний редактор макетів;

— можливість взаємодії компонетів інтерфейсу користувача;

— можливість переглядати макети одразу у кількох конфігураціях екрану;

— подання підсистеми, що розробляється, онлайн при створенні;

— користування консолю розробника: покращені підказки, підтримка перекладу, відстеження напрямку;

— набір підсистем, які були створені за допомогою Gradle;

— генерація типів збірок та генерація файлів із розширенням .apk;

— здатність покращити написаний код, використовуючи функцію швидкого

виправлення у коді;

— статичний аналізатор коду, за допомогою якого можна виявляти помилки продукту, або несумісність версій;

— вбудована утиліта для підпису створених підсистем;

— макети та шаблони компонентів Android;

— вбудований редактор макетів, що дозволяє користувачу розмістити компоненти інтерфейсу користувача;

— підтримка створення підсистем на Android TV та Android Wear;

— підтримка Google Cloud Platform, в тому числі інтеграція сервісів App Engine;

— Android Studio 2.2 підтримує функцію, за допомогою якої розробник створює підсистеми для нової платформи;

— версія Android Studio 2.2 співпрацює із компілятором Jack;

— підтримується за допомогою Java 8;

— Android Studio 3.0 включає мовні засоби програмування Kotlin.

Щоб розробити підсистему потрібно встановити Android SDK. Якщо його не було завантажено одразу, то середовище розробки Android Studio, видасть помилку, яка повідомить, що Android SDK не є встановленим у Android Studio.

Під час першого використання Android Studio, необхідно створити новий проект. Натисніть «Створити новий проект Android». Впишіть назву підсистеми, яку ви збираєтесь створити та доменне ім’я підсистеми. Інформація буде використана, щоб розмістити файли підсистеми у системі файлів Android. Наступний етап створення підсистеми — обирання платформи і мови програмування. Варто використовувати Android смартфони версії 4.3+, так як це найпоширеніші версії, після чого оберіть мову Java.

Залишається лише обрати дизайн підсистеми. Шаблоном основного вікна є «Activity». Підсистема може не мати вікна «Activity», але рекомендується його використовувати. Інтерфейс середовища розробки схожий на всі схожі середовища

розробки. Створення підсистеми на платформу Android відрізняється від програмування, тому що вам потрібно об’єднати кілька файлів ресурсів.

Програмування Android Studio створено у файлі «Java», який має таку же назву, як і Activity файл. Дизайн підсистеми розташований у файлі «xml», де на мові розмітки описано компоненти підсистеми. Після створення кнопки потрібно охарактеризуваи її у файлі «xml», після чого додайте до неї функції, за допомогою файлу «java».

У таблиці 3.1 наведено системні вимоги щодо використання інтегрованого сереередовища Android Studio.

Таблиця 3.1 — Системні вимоги щодо використання інтегрованого середовища Android Studio

Характеристика	**Windows**	**OS X**	**Linux**
Версія ОС	Microsoft Windows 7/8/10/11 (64-bit)	MacOS X 10.9.+	GNOME або KDE
Оперативна пам'ять	4 або 8 ГБ		
Вільний простір	2 GB вільного простору, 4 GB для SDK		
Java	Java Development Kit 9		
Розширення екрану	1280 x 720		

Таким чином, користуйтеся одним эдинм макетом для кількох Activity. Дія Activity може обробляти декілька XML-файлів описів зображення. У будь-якому випадку необхідно отримати дозвіл на керування усіма файлами проекту. За допомгою вікон зліва можна переходити від одного файлу до іншого, головне тримати їх відкритими.

Шаблон головного вікна створення проекту IDE Android Studio зображено на рисунку 3.1.

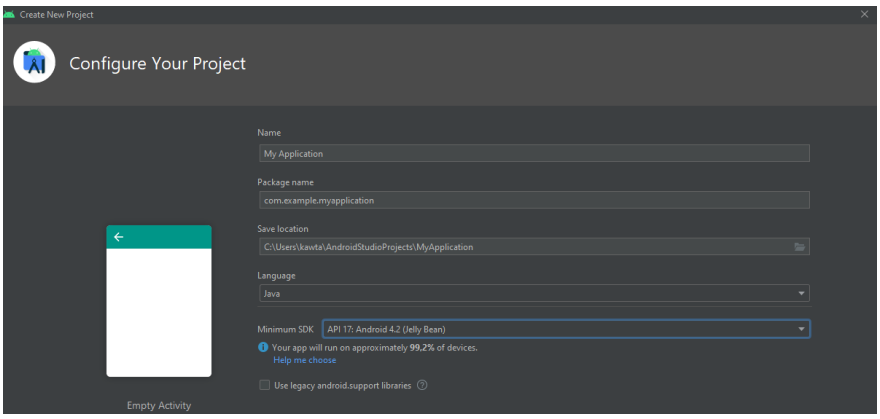

Рисунок 3.1 — Шаблон головного вікна створення проекту IDE Android Studio

Android Studio створює папку «res», у якій розміщені ресурси та папка «values», у якій збережені XML-файли різних значень змінних. Назви кожноо файлу мають бути написані малими літерами. Головна інформація підсистеми, що створюється зберігається у файлі «AndroidManifest.xml», у якому описані дозволи та назва програми [13].

Під час створення підсистеми, можливо спрограмувати будь-які файли та Activity, щоб розширити функціональність підсистеми, що створюється. Натисніть на праву кнопку миші, та оберіть функцію «Create».

Редагуючи файли XML, нижче створюються вікна «Design» і «Text». У них можливо описати код власноруч, та можливо створити необхідні функції графічного режиму. Вікні «Палітра» необхідно, щоб створити тип панелі. Щоб додати панель, перетягніть її до макету програми.

Коли підсистема буде розроблена, її необхідно протестувати, можна зробити запустивши її на мобільному пристрої або використавши емулятор. Після чого запустити створену підсистеми буде просто. Підключіть пристрій до USB, після чого клацніть на кнопку «Запустити програму». У налаштуванні пристрою потрібно включити «налагодження по USB» та параметр «встановлення із недостовірних джерел». У вікні, що відкрилося, необхідно обрати ваш гаджет або

створити віртуальний. У вікні, що відкрилося, потрібно обрати свій пристрій, після чого підсистема буде запущеною на ньому.

Синя рамка справа змінює положення та розмір відкритих елементів (зображено на рисунку 3.2).

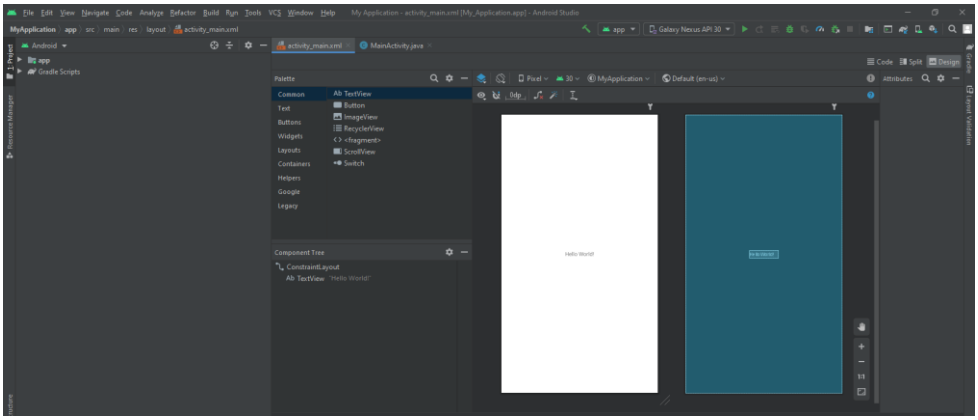

Рисунок 3.2 — Вкладки «Текст» та «Дизайн» в Android Studio

У вікні віртуального пристрою можливо змінити розмір екрану, дизайн і функціональність для будь-яких гаджетів Android. Далі необхідно включити компоненти підсистеми. Зайдіть у меню «Інструменти» — «Android» — «AVD Manager».

Елементи та інструменти SDK потрібно використовувати, якщо підсистема розробляється для необхідного пристрою Android. Їх розташування знаходиться на панелі «Інструменти» – «SDK Manager». Там можливо встановити репозиторій для Android. Просто поставте прапорці поруч із елементами, які необхідно завантажити та натисніть кнопку «ОК» [14].

Коли тестування розроблювальної підсистеми буде завершено, завантажте створену підсистему у магазин «Google Play Market», для цього завершивши створення файлу .apk. У меню «Інструменти» натисніть «Створити підписаний APK». Для цього створіть та оберіть сховище ключів. Перед вами з’явиться сертифікат автентичності, що підтвердить створення підсистеми саме вами. Виконайте захищення облікового запису Google Play Market від хакерів і запобіжіть можливість зберігання невірних файлів із розширенняи apk. Збережіть створений

файл на портативному комп’ютері, але якщо файл буде втрачено, оновлення підсистеми стане неможливим. Оберіть «Випуск файлу», у типу збірки. Після чого видаліть небажаний вміст із файлу APK, та натисніть кнопку «Заершити».

3.2 Обґрунтування вибору серверу бази даних

База даних Firebase — хмарна база даних класу NoSQL, яка дозволяє розробникам підсистеми зберегти та виконати синхронізацію, яка збережена для декількох клієнтів за допомогою простих інструментів та методів зв’язків між ними.

Firebase надає базу даних у реальному часі, що допомагає розробнику підсистеми API синхронізувати дані між користувачами, після чого зберегти їх у хмарному сховищі Firebase. Формат збереження JSON синхронізує у режимі реального часу з кожним клієнтом, що виконав підключення [15].

Firebase забезпечує клієнтськими бібліотеками, які інтегруються із підсистемами Android, Swift, IOS, JavaScript та Java. Доступ до бази даних Firebase виконується за допомогою REST API, що зв’язує створені сценарії JavaScript, а саме AngularJS, Backbone.js, React та Ember.js. Використавши функцію Realtime Database ви зможете зберегти особисті дані, використавши вказані правила безпеки, що були використані на сервері.

Функції бази даних Firebase:

— робота в режимі реального часу часі;

— робота автономного режиму;

— синхронізація із смартфонами користувачів;

— масштабування бази даних.

Firebase синхронізує дані не використовуючи HTTP-запити, тому під час змінення даних, будь-який пристрій, який є підключеним до неї оновиться за декілька мілісекунд.

Підсистеми, які створені у базі даних Firebase є надійними у автономному режимі, тому що Firebase завантажує збережені дані з диску. Коли з’єднання буде

відновлено, тоді пристрій користувача зможе отримати інформацію про внесені зміни, виконавши синхронізацію із станом сервера на дану секунду.

Отримання доступу до бази даних Firebase відбувається із мобільного пристрою користувача або браузера, отже не потрібно створювати окремий сервер для доступу підсистемі. Надійність та перевірка наявності даних виконується у правилах безпеки бази даних Firebase на основі регулярних виразів, що досягаються під час зчитування та запису даних.

Firebase Database виконує усі вимоги масштабування підсистеми в даних, для цього вона розділяє дані між декількома екземплярами бази даних. Використовуйте автентифікацію Firebase для створеного проекту, і тоді ви зможете перевіряти користувачів у екземплярах бази даних. Контролюйте доступ до даних у базі даних, використовуючи спеціальні правила Firebase для кожного його екземпляра.

Створіть проект у Firebase Console, для цього необхідно перейти на головну сторінку консолі. На екрані з’являться два варіанти:

— новим користувачам бази даних, запропонують набрати вітальне повідомлення та запрошення на створення нового проекту;

— якщо ви вже стоврювали проекти у Firebase, відкриється список та біла картка додавання проекту.

Залежно від вибраних параметрів, натисніть на кнопку для створення проекту. Початковий екран запропонує написати назву для вашого проекту. Після чого будете використано те саме ім'я, що має створена підсистема, після збереження натисніть на кнопку «Продовжити».

Створивши проек, розпочнеться запуск початкового екрану. Завантаження відбуватиметься деякий час. Після завершення процесу значок завантаження стане у вигляді трьох помаранчевих точок, у якому при натисненні зобразиться текст підтвердження та кнопка «Продовжити». Натиснувши на неї, база даних переведе користувача на основний екран проекту (зображено на рисунку 3.3).

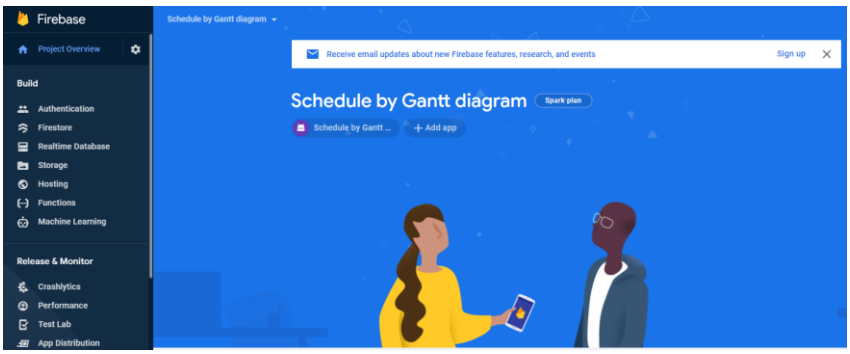

Рисунок 3.3 — Основний екран проекту бази даних Firebase

Консоль бази даних має три функції, а саме налаштування проекту, автентифікація та база даних. У меню зліва є іконка, у якій треба вибрати «Налаштування проекту». Там знадобиться відкрити конфігураційні дані в налаштуванні проекту. Створені дані треба підключити до підсистеми, що розробляється. Перед тим як додати створений код до проекту, необхідно налаштувати Firebase у створеній підсистемі. Вставте конфігураційний код та виконайте запускання методу firebase.initializeApp(), у якому потрібно записати об'єкт конфігурації в параметрі цього методу. Операція повинна виконуватись, коли запущена ініціалізація програми.

Для того щоб зареєструватись або авторизуватись, необхідно повернутися до консолі Firebase та перейти на сторінку «Автентифікація». Завантажаться вкладки «Користувач» та «Спосіб реєстрації». Усі зареєстровані користувачі підсистеми будуть відображені на вкладці «Користувач». Вкладка «Спосіб реєстрації» надає можливість створити та налаштувати спосіб, який буде зручний користувачу, також є можливість включення реєстрація за адресою електронної пошти та введеним паролем. Перейдіть на вкладку «Адреса електронної пошти та пароль» та огляньте збережені налаштування.

Для того щоб додати базу даних до підсистеми, необхідно на основному екрані проекту натиснути на значок Android білого кольору, після чого розпочнеться процес інтеграції. Спочатку Firebase запитає дані створеної підсистеми. Ввівши файли підсистеми, сгенерується JSON-файл «google-

services.json». Натисніть на кнопку «Проект», після чого відкриється дерево файлів. Виконайте завантаження файлу із консолі бази даних на комп'ютер, після чого додайте файл в «app». Більшість часу ви працюватимете з проектами підсистем для Android, тому що воно є найлегшим способом навігації, коли буде писатися код підсистеми.

Для того, щоб завершити налаштування консолі бази даних, потрібно повернутися у консоль Firebase та натисніть «Далі». Це необхідно щоб додати файл JSON та зв’язки у створений проект. Внизу екрана зобразиться примітка, про те, що Firebase перевіряє з’єднання між консоллю та створеною підсистемою. Далі необхідно побудувати та запустити створену підсистему. Firebase має підтвердити з’єднання відобразивши кнопку «Продовжити у консолі».

Після успішного з’єднання з’явиться фіолетовий значок із логотипом Android на основному інтерфейсі проекту, що означатиме підключення створеного проекту до підсистеми.

Створення таблиці Firebase наведено на рисунку 3.4.

```
public class MainActivity extends AppCompatActivity {
    Button button_sign_in, button_register_now;
    FirebaseAuth auth;
    FirebaseDatabase db;
    DatabaseReference users;
    RelativeLayout root;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        button_sign_in = findViewById(R.id.button_sign_in);
        button_register_now = findViewById(R.id.button_register_now);
        root = findViewById(R.id.root_element);
        auth = FirebaseAuth.getInstance();
        db = FirebaseDatabase.getInstance();
        users = db.getReference( path: "Users");
        button_register_now.setOnClickListener(v -> showRegisterWindow());
        button_sign_in.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) { showSignInWindow(); }
        });
    }
```

Рисунок 3.4 — Створення таблиці та підключення підсистеми до Firebase

У підсистемі після підключення до Firebase потрібно внести назву бази даних

та коректну версію. Що обробити підключення до Firebase, потрібно знайти абстрактний метод OnCreate() у класі AppCompatActivity, який є нащадком MainActivity. Метод буде викликаний, якщо база даних не буде створеною. Метод зворотного виклику onCreate() реалізовує алгоритм для того, щоб створити таблицю.

3.3 Проектування підсистеми коригування термінів вивчення обраних дисциплін системи дистанційного навчання

3.3.1 Розробка архітектури підсистеми

Архітектура інформаційних технологій визначає основні елементи мережі, описує її загальну логічну структуру, технічні компоненти, частини програмного забезпечення та методи кодування, а також визначає принципи взаємодії із інтерфейсом користувача.

Архітектура «клієнт-сервер» — концепція мережі інформації, де головна частина ресурсів буде зосередженою саме на сервері, що буде обслуговувати клієнтів [16].

Ця архітектура є одним із структурних шаблонів програмного забезпечення і популярна при створенні розподілених мережевих підсистем, що передбачають взаємодію та обмін даними між ними.

«Клієнт-сервер» складається з таких компонентів:

— сервер, що надає інформацію, для того щоб звернутися до них;

— сукупність клієнтів, що користуються послугами, що надаються сервером;

— мережу, яка забезпечує взаємодію між клієнтами та серверами;

— дані.

Основна роль сервера полягає в управлінні та обслуговувані запитів від клієнтів, які спільно використовують системні ресурси в певний момент часу. За своїми функціями сервери поділяють на файлові сервери, обчислювальні сервери, сервери друку, сервери зв’язку тощо.

Одночасний доступ багатьох користувачів до інтегрованої бази даних

досягається в архітектурі «клієнт-сервер», де сервер відіграє більш активну роль. Запит на обробку даних надсилається клієнтом на сервер через мережу, а сервер шукає та обробляє дані через систему керування базами даних. Оброблені дані передаються від сервера до клієнта. Унікальною особливістю цією архітектури є використання мови структурованих запитів SQL (Structured Queries Language), яка дозволяє обробляти загальні дані для різних додатків у мережі.

Сервер — елемент обладнання, що дозволяє спільно використовувати послуги в мережі; програмний компонент, який забезпечує належне функціонування послуги; поєднання апаратних та програмних компонентів.

Клієнт — комп'ютер; програма, яка обробляє, формує та передає запити на сервер. Сервер приймає запит, обробляє його і відправляє результат клієнту. Користувачі взаємодіють лише з клієнтом. Крім того, клієнтські програми та їхні запити зберігаються окремо від системи керування базами даних.

Серверна та клієнтська частини незалежні. Загальноприйнято вважати, що клієнти та сервери — переважно програмні модулі. Здебільшого вони знаходяться на різних комп'ютерах, але бувають випадки, коли обидві частини фізично знаходяться на одній машині; у цьому випадку сервер часто називають локальним [17].

Технологія «клієнт-сервер» базується на таких фактах:

— усі публічні дані користувача знаходяться в одному або декількох серверах;

— клієнти спільно (паралельно та одночасно) обробляють спільні дані.

Тобто системи, засновані на цій технології, розділені для користувачів, тому їх зазвичай вважають різновидом багатокористувацьких систем [18].

Переваги архітектури «клієнт-сервер»:

— можливість створення мереж з великою кількістю робочих станцій;

— забезпечує централізоване управління обліковими записами користувачів, доступом та безпекою, що спрощує керування мережею;

— забезпечення ефективного доступу до ресурсів мережі;

— дозволяє створювати спільні паролі для входу в мережу та отримання доступу до всіх ресурсів.

Недоліки архітектури «клієнт-сервер»:

— збої сервера призводять до простою та втрати ресурсів мережі;

— обслуговування здійснюється кваліфікованим персоналом;

— висока вартість мережі та мережевого обладнання.

Підсистема для врахування змін у термінах вивчення обраних дисциплін на основі діаграми Ґанта складається з двох частин: сервера та клієнта.

3.3.2 Розробка бази даних підсистеми

Перш ніж створювати таблиці, форми та інші об'єкти для свого сайту, необхідно створити базу даних (БД). Відображення описів предметних областей у схемі внутрішньої моделі даних є складним процесом при проектуванні бази даних. Цей процес є серією простіших процесів розробки менш складних відображень. Ця послідовність у процесі проектування постійно уточняється та вдосконалюється, щоб визначити об'єкти, їхні властивості та зв'язки, які знадобляться майбутнім користувачам системи.

Проектування бази даних — ітеративний багатоетапний процес, у якому обґрунтовані рішення приймаються під час аналізу інформаційної моделі предметної області, потреб програмістів та користувачів, синтезу фізичних та логічних структур даних, аналізу та обґрунтування вибору програмного забезпечення [19]. Типи представлення даних при проектуванні бази даних: зовнішнє, інформаційне, логічне (логіка даних), внутрішнє.

Зовнішній рівень: необхідно визначити функціональність керованого об’єкта, для якого створюється база даних, всю вихідну та отриману інформацію з точки зору визначення потреб зберігання даних. Зовнішнє представлення міститься на інформаційному рівні, де будується інформаційна (канонічна) модель даних, яка є не просто сумою зовнішніх інтерпретацій даних.

Інформаційний рівень — інформаційно-логічна модель (ІЛМ) предметної

області, в якій виключена надлишковість даних та відображені інформаційні характеристики керованих об’єктів незалежно від характеристик та деталей бази даних. Тобто цей рівень в першу чергу стосується користувачів, які розробляють або застосовують бази даних.

Логічні (концептуальні) рівні створюється з урахуванням деталей та характеристик бази даних. Цей рівень зосереджений на комп’ютерній обробці та програмістах, які її виконують. Тут створюється концептуальна модель даних, тобто впорядкована модель предметної області, яка враховує особливості та обмеження бази даних.

Внутрішній рівні пов’язані з фізичним розміщенням даних у пам’яті комп’ютера [20]. Тут будується фізична модель бази даних, що містить дані про опис форматів записів, їх порядок, розміщення за типом пристрою, характеристики та способи доступу до даних. Обсяг пам'яті та час відгуку системи залежить від параметрів фізичної моделі.

Етапи проектування бази даних.

Встановлення цілі. Встановлення призначення, інформації та основних функцій, яку вони повинні нести. База даних повинна відповідати вимогам користувача.

Визначення таблиці. При складанні таблиць слід керуватися такими основними принципами: інформація в таблицях не повинна повторюватися, кожна таблиця містить дані лише одного суб’єкта.

Визначення обов’язкових полів. Тут потрібно враховувати, що будь-яке поле має бути пов’язане з темою таблиці. Не бажано додавати до таблиці дані, отримані під час обчислень. Таблиця повинна містити всі необхідні дані, бажано розбиті на найменші логічні одиниці.

Призначення індивідуальних значень для всіх полів.

Визначення зв’язку між таблицями. При розподілі інформації в таблицях і визначенні ключових полів необхідно вибрати типи інформаційного зв’язку між таблицями.

Перевірка бази даних на наявність помилок і дублікатів.

Додавання інформації для формування інших об’єктів бази даних. Якщо структура таблиці відповідає вимогам, можна вводити всі дані.

3.3.3 Розробка інтерфейсу підсистеми

Проектування підсистем та розробка інтерфейсів підсистеми є важливим етапом, для того щоб створити програмний продукт. Ця робота залежить від того, як користувач сприйматиме створену підсистему, чи сподобається вона йому, чи зручною буде підсистема у застосуванні та чи стане вона популярною.

Інтерфейс підсистем для Android відіграє дуже важливу роль у процесі створення, він є механізмом зв’язку апаратної та програмої частини забезпечення мобільного пристрою, оскільки він є центром зв’язку із користувачем.

Основою метою будь-якої підсистеми є забезпечення максимальної зручності або ефективності обробки інформації. Інтерфейс підсистеми є важливою частиною під час створення.

Конструкція мобільної підсистеми повинна бути зрозумілою та простою. Так як екран мобільного телефону не є великим, в порівнянні з монітором персонального комп'ютера, неможливо дотримуватися тих самих правил, що й для комп'ютера при розробці підсистеми для мобільних пристроїв.

Правильно розроблена мобільна підсистема повинна поєднувати у собі наступні три основні властивості. Створена підсистема матиме грамотний та правильний дизайн, інтеграцію та використовуватиме усі можливі функції для мобільних телефонів. Наступним атрибутом є важливість зацікавлення користувача у створеній підсистемі. Найкращий спосіб змусити людину користуватися підсистемою з великою кількістю функцій — додати мотивацію та зацікавлення для користувача. Останнім атрибутом є практичність, тому що високі рейтинги оцінення мають ті підсистеми, які дійсно є необхідними. Під час розробки підсистеми дотримувалися усі правила, тому було створено гарну та зрозумілу підсистему коригування термінів [21].

Створений код розмітки MainActivity наведений на рисунку 3.5.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id = "@+id/root_element"
    android:background="@drawable/whtbg"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="25dp"
        android:layout_centerHorizontal="true"
        android:gravity="center"
        android:orientation="vertical">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:fontFamily="@font/suez_one"
            android:gravity="center_horizontal"
            android:text="Врахування змін у термінах вивчення дисциплін"
            android:textAlignment="center"
            android:textColor="@android:color/black"
            android:textSize="35sp" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:fontFamily="@font/montserrat_bold"
            android:gravity="center_horizontal"
            android:text="на основі діаграми Ґанта"
```

Рисунок 3.5 — Розмітка створеного класу MainActivity

Щоб зв’язати створену розмітку, потрібно користуватися написаним кодом у методі onCreate(): setContentView(R.layout.activity_main), де «main» — назва основної розмітки створеної підсистеми.

На усіх інших вікнах підсистеми, без основної сторінки наявна кнопка «Профіль», де користувач підсистеми може у будь-який момент перейти до вікна розкладу, видалити вивчений або непотрібний предмет для вивчення та завершити роботу у своєму акаунті. За допомогою кнопки «Повернутись» користувач повертається до минулого вікна створеної підсистеми. Після авторизації у підсистемі користувач потрапляє у вікно, де йому необхідно обрати день тижня, після чого він повинен створити предмети для вивчення.

Лістинг вікна «MainActivity» наведено у додатку Б.

На рисунку 3.6 зображено вікно підсистеми «Початкова сторінка підсистеми».

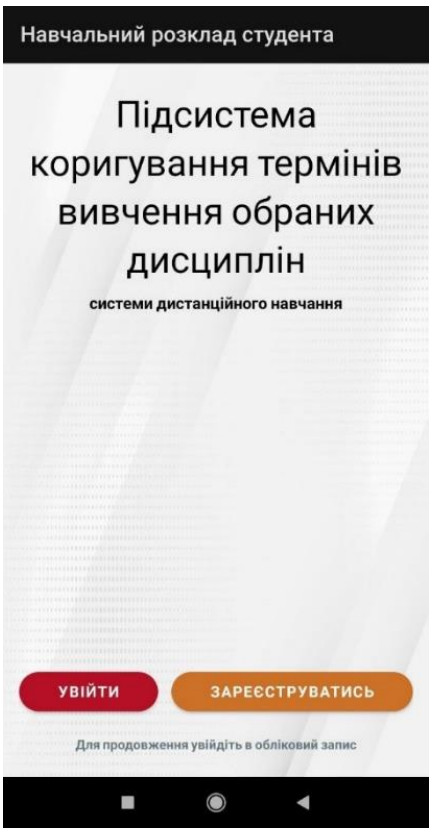

Рисунок 3.6 — Початкова сторінка підсистеми

Лістинг вікна «SubjectActivity» наведено у додатку В.

Код розмітки профілю користувача та початку реєстрації наведено на рисунках 3.7 та 3.8 відповідно.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#A1A2AA"
    app:cardBackgroundColor="@color/design_default_color_background"
    app:cardElevation="10dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:orientation="vertical">

        <com.rengwuxian.materialedittext.MaterialEditText
            android:id="@+id/username"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Логін користувача:"
            android:inputType="text"
            android:text=""
            android:textColor="@color/design_default_color_primary"
            android:textColorHint="@color/design_default_color_primary_dark"
            android:textSize="20sp"
            app:met_floatingLabel="highlight"
            app:met_primaryColor="#982360"

            app:met_singleLineEllipsis="true" />

        <com.rengwuxian.materialedittext.MaterialEditText
            android:id="@+id/email"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="E-mail:"
```

Рисунок 3.7 — Розмітка початку реєстрації користувача у підсистемі

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:cardElevation="10dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:orientation="vertical">

        <Button
            android:id="@+id/returnToSchedule"
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="1"
            android:padding="15sp"
            android:background="@drawable/button_sign_in"
            android:fontFamily="@font/montserrat_bold"
            android:text="Перейти до розкладу"
            android:textColor="@android:color/white"
            android:textAlignment="center"
            app:backgroundTint="@color/button_sign_in_background"
            android:gravity="center_horizontal"
        />

        <com.google.android.material.button.MaterialButton
            android:id="@+id/exitAccount"
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="1"
            android:padding="15sp"
            android:layout_marginTop="50dp"
```

Рисунок 3.8 — Розмітка створеного профілю користувача підсистеми

Підсистема коригування термінів вивчення обраних дисциплін системи дистанційного навчання створена у вигляді простого та зручного інтерфейсу. Головна сторінка підсистеми містить назву, дві кнопки «Увійти» та «Зареєструватись».

На рисунку 3.9 зображено вікно створених предметів для вивчення обраного дня тижня. У вікні предметів студент може додати необхідну кількість предметів для вивчення.

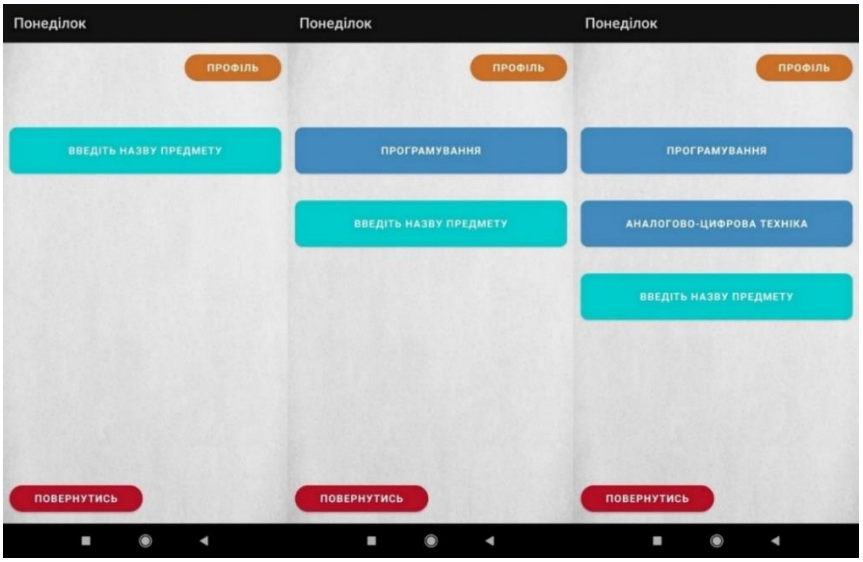

Рисунок 3.9 — Створення предметів для вивчення обраного дня тижня

Activity реалізовано за допомогою розмітки LinearLayout, так як у нашому випадку її найзручніше використовувати.

Створивши предмет для вивчення, студент потрапляє на нове вікно підсистеми, де він повинен створити необхідну йому кількість тем для вивчення обраного предмету. На рисунку 3.10 зображено вікно тем обраного предмету для вивчення.

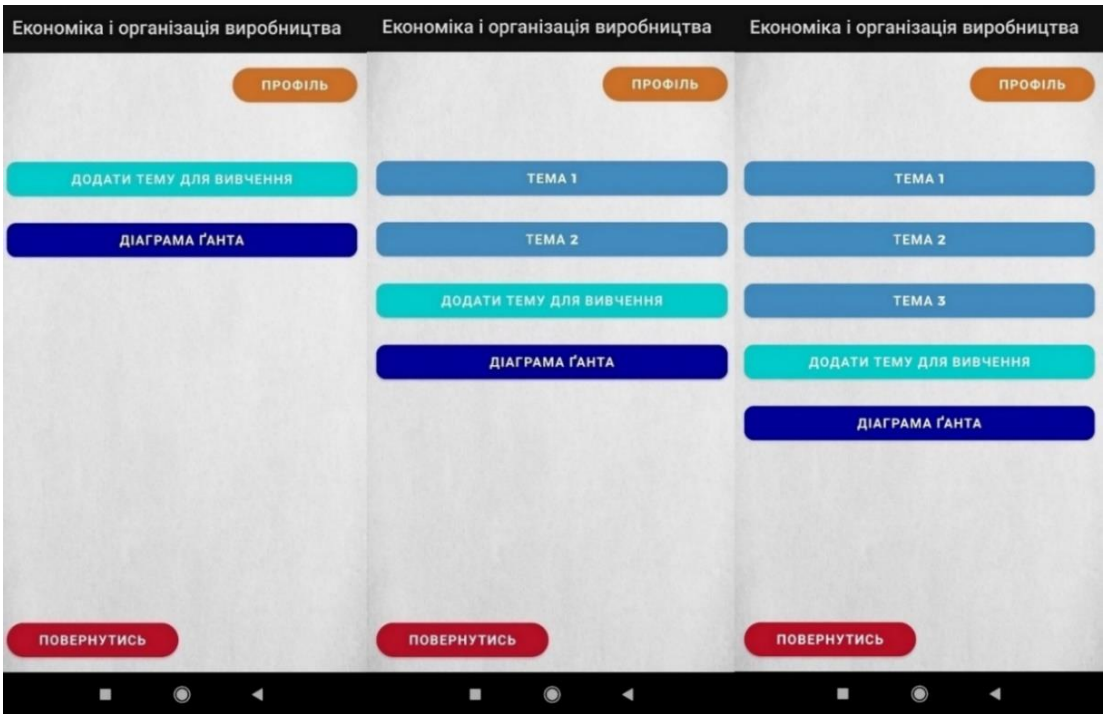

Рисунок 3.10 — Вікно розробленої підсистеми, де студент створює теми створеного обраного предмету для вивчення

Лістинг вікна «SubjectTopicActivity» наведено у додатку Г.

3.4 Тестування підсистеми та аналіз результатів

Тестування програмного забезпечення — процес технічного дослідження, яке проводиться на вимогу клієнтів для визначення характерної інформації підсистеми, яка пов’язана параметрами, щоб використовувати створений продукт, або є процесом аналізу або використання програмного забезпечення, щоб виявляти дефекти.

Тестування прогнозує роботу або аналіз підсистеми, що розробляється. Статичне тестування — тестування, яке є пов’язане з аналізом результатів розробки програмного забезпечення. Статичне тестування передбачає перегляд програмного коду, контроль та перевіряє створену підсистему. Динамічне тестування — тестування, яке передбачає функціональність програмного продукту. Статичне та динамічне тестування є взаємопов’язаними.

Тестування програмного коду — процес цілеспрямованого виконання коду програми, для виявлення наявних дефектів. Дефект є частиною коду програми, обробка якої викликає неочікувану поведінку пристрою. Збої можуть бути викликані неочікуваною поведінкою підсистеми, в даному випадку значними недоліками коду програми. Невеликі помилки у коді викликають проблеми, які не будуть порушувати роботу підсистеми, але ускладнюють її використання є загальними або незначними дефектами.

Метою використання порядкового тестування програмного коду є зменшення наявності помилок у створеному продукті. Тестування не гарантує, що системний код вільний від дефектів. Якщо поєднати перевірку із процесом валідації та верифікації, які спрямовані на видалення будь-яких помилок та неповної проектної документації, тоді гарно організоване планування буде гарантувати, що підсистема буде відповідати вимогам та працюватиме у будь-якій очікуваній ситуації [22].

В ході тестування перевірялися наступні складові комп’ютерної програми:

— реєстрація та авторизація студента;

— коректність зображення діаграми Ґанта, що зображує запланований час користувача та дійсний витрачений час на вивчення кожної обраної теми створеного предмету.

Перевіримо, чи справді реєстрація нового користувача виконана без помилок. Перейдемо на сторінку бази даних Firebase, де необхідно відкрити базу даних реєстрації у розробленій підсистемі, яка зображена на рисунку 3.11.

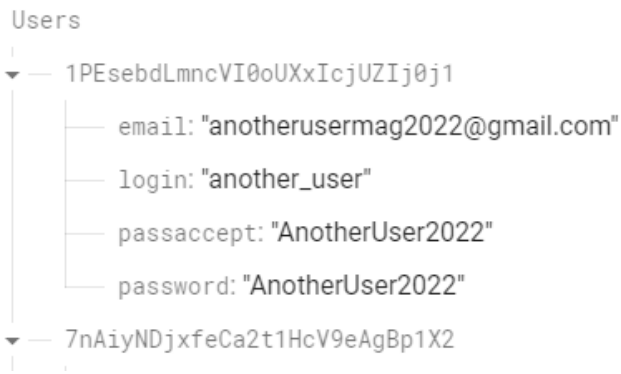

Рисунок 3.11 — Успішна реєстрація у базі даних Firebase

Після запуску підсистеми, користувач входить у створений ним акаунт або проходить процес реєстрації, також він може просто закрити підсистему. Виконаємо тестування процесу реєстрації нового користувача у розробленій підсистемі та поглянемо, чи будуть введені дані записані у базу даних Firebase. У вікні реєстрації вводимо логін користувача «another_user», електронну пошту «anotherusermag2022@gmail.com» та пароль «AnotherUser2022». Вікно реєстрації нового користувача у створеній підсистемі з введенними даними зображено на рисунку 3.12.

Рисунок 3.12 — Реєстрація нового користувача у створеній підсистемі

Оскільки діаграма Ґанта будується у режимі реального часу, існує можливість подивитись на будування графіка та зміщення запланованих термінів у режимі реального часу.

Лістинг вікна «GanttDiagramActivity» наведено у Додатку Д.

Після вивчення обраних тем створеного вами предмету, потрібно повернутись до вибору теми, де натискаємо на кнопку «Діаграма Ґанта». Якщо студент дійсно вивчив усі обрані теми створеного предмету, в такому випадку буде побудована діаграма Ґанта вивченого предмету, із порівняннями запланованих та дійсних термінів вивчення кожної теми обраного предмету. Синім кольором позначено заплановані терміни, червоним кольором — реальний час вивчення обраної теми, який перевищує запланований час, та зеленим кольором — реальний час, коли студент справився із вивченням обраної теми швидше за запланований термін. Декілька прикладів із діаграмою Ґанта різних предметів зображено на рисунку 3.13.

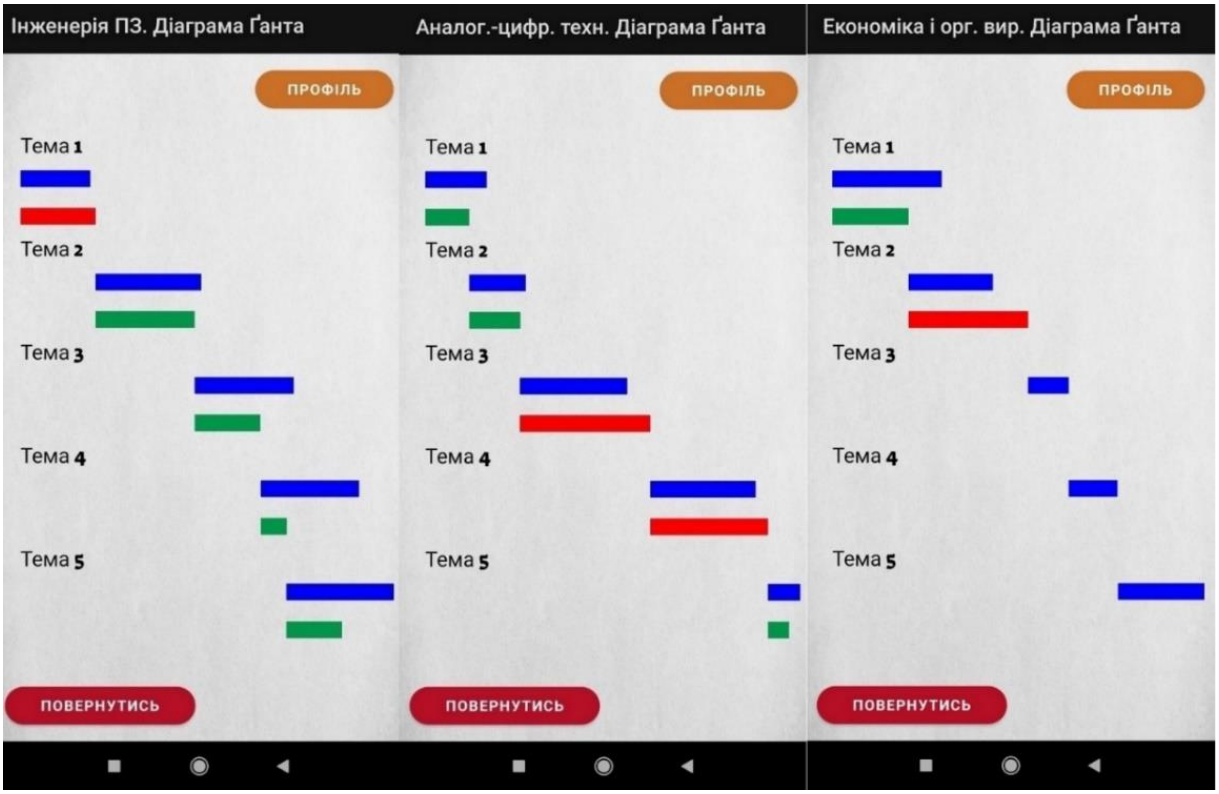

Рисунок 3.13 — Побудована діаграма Ґанта предметів для вивчення

3.5 Інструкція користувача

Підсистема надає змогу користувачу увійти у свій створений профіль, де йому необхідно ввести дані для термінів вивчення обраних тем кожного створеного користувачем предмету.

Увійшовши перший раз у підсистему, вам необхідно буде зареєструватись у ній, щоб користуватись нею і надалі. Під час реєстрації логін та електронна пошта повинні бути введеними, оскільки з’являться помилки «Логін користувача не введено» та «Електронна пошта користувача не введена». Число символів паролю користувача має перевищувати 4 символи, тому що якщо їх буде менше, тоді зобразиться помилка «Було введено малу кількість символів паролю користувача». Паролі, що були введені користувачем у полі «Пароль» та «Підтвердіть пароль» мають бути однаковими, тому шо якщо хоча б одна буква або цифра не будуть співпадати, тоді буде видано помилку «Паролі користувача не співпадають».

Після успішної реєстрації у підсистемі, користувачу потрібно авторизуватись, щоб користуватись створеною підсистемою. Коли користувач авторизовується, то поля з електронною поштою та паролем, так як і при реєстрації користувача, повинні бути заповненими. Під час введення даних авторизації, перегляньте, чи вони дійсно є вашими даними, які були введені вами під час реєстрації користувача, оскільки при невірних даних ви не зможете використовувати підсистему, тому що буде видана помилка «Помилка в авторизації користувача».

У необхідний вам час, можливо повернутись назад, для цього просто натисніть на кнопку «Повернутись». Під час натискання на кнопку «Профіль» ви переходите на спливаюче вікно, де можливо повернутись до розкладу навчання, видалити будь-який створений вами предмет для вивчення або завершити роботу у підсистемі повернувшись на головну сторінку підсистеми, де потрібно буде починати все спочатку. Вікно профілю користувача зображено на рисунку 3.14.

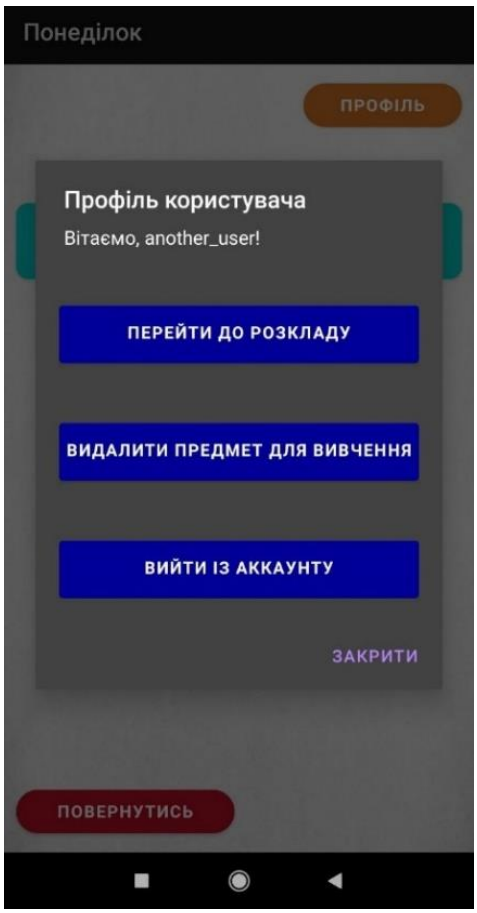

Рисунок 3.14 — Вікно профілю користувача у створеній підсистемі

Після авторизації у підсистемі, ви перейдете до вибору дня тижня, що зображено на рисунку 3.15. Оберіть день тижня, щоб розпочати створення предметів для індивідуального вивчення. Обравши день тижня, наступним вашим кроком повинно бути створення предмету навчання, тому натисніть на кнопку «Введіть назву предмету», та впишіть назву предмету, після чого натисніть кнопку «Зберегти». Оберіть створений предмет для вивчення — вас перекине у нове вікно де вам необхідно буде створити необхідну для вас кількість тем для вивчення обраного предмету, після чого перейдіть у кожне вікно створених тем предметів та введіть запланований час їх вивчення. Коли ви будете готові розпочати навчання натисніть на кнопку «Старт» для початку відліку реального часу вивчення. Згідно моделі навчання варто починати ваше навчання з першої теми, тоді вивчення обраних тем буде ефективнішим. Після завершення вивчення обраної вами теми створеного предмету, натисніть кнопку «Фініш».

Рисунок 3.15 — Вікно вибору дня тижня для вивчення дисциплін у підсистемі

Перейшовши до наступної обраної вами теми створеного предмету, перегляньте чи були зміненими заплановані терміни вивчення дисципліни. Якщо вивчення попередньої теми було завершено раніше за вказані терміни, тоді час, який залишився додасться до наступної теми того самого предмету навчання. Якщо вивчення попереньої теми було завершено пізніше за вказані терміни, тоді запланований час вивчення наступної теми того самого предмету навчання зменшиться на увесь лишній час, що був використаний для вивчення попередньої обраної теми предмету. Далі алгоритм не змінюється для усіх інших обраних тем.

Завершвши вивчення обраної вами теми створеного предмету для вивчення, натисніть на кнопку «Фініш». Після чого у вікні вивченої вами теми зобразиться реальний час вивчення обраної теми, як зображено на рисунку 3.16.

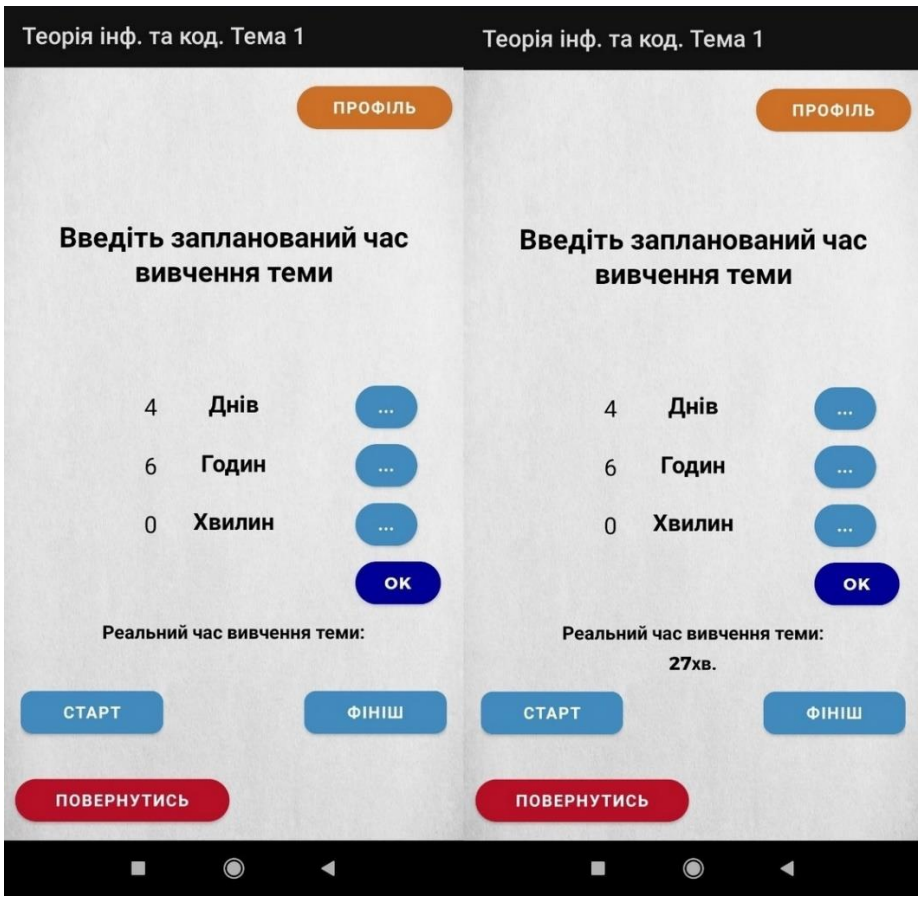

Рисунок 3.16 — Реальний час вивчення обраної теми користувача у створеній підсистемі

Після завершення вивчення усіх обраних тем створеного предмету, поверніться до обирання теми, натисніть на кнопку «Діаграма Ґанта», після чого буде зображено діаграму Ґанта вивченого створеного предмету, де буде виконано порівняння запланованих та дійсних термінів вивчення кожної обраної теми даного створеного предмету.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку нового програмного продукту підсистема коригування термінів вивчення обраних дисциплін системи дистанційного навчання. Завдяки створеній підсистемі з'являється можливість ефективного безперервного планування та прийняття рішень у режимі реального часу. Особливістю програми є те, що підсистема працює за власними математичними моделями, завдяки яким студент вказує запланований час вивчення обраної теми, а підсистема після її вивчення підраховує реальний витрачений час та автоматично змінює запланований час вивчення наступної теми в залежності від успішності та витраченого часу студента. Аналогом може бути комп'ютерна програма «TeamGantt»: Lite-версія — 19$ в місяць; Pro-версія — 49$ в місяць; Enterprise-версія — 99$ в місяць. Програмний продукт «Microsoft 365»: для одного користувача — 1899 грн. в рік; для 2-6 користувачів — 2599 грн. в рік.

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п'ятибальної системи оцінювання за 12-ма критеріями, у відповідності із таблицею 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Крите-рій	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздат-ність в реаль-них умовах

Продовження таблиці 4.1

Бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
Ринкові переваги					
2	Багато аналогів на малому ринку	Ринкові п Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною .	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практик на здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так із комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування

Продовження табл. 4.1

Бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Усі дані по кожному параметру занесено в таблиці 4.2.

Таблиця 4.2 – Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	**ПІБ експертів**		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	3	3	3
Наявність аналогів на ринку	3	3	4
Цінова політика	4	4	4
Технічні та споживчі властивості виробу	4	3	3
Експлуатаційні витрати	3	4	3
Ринок збуту	4	3	4
Конкурентоспроможність	3	3	3

Продовження табл. 4.2

Критерії оцінювання	**ПІБ експертів**		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Фахівці з технічної і комерційної реалізації	4	3	4
Фінансування	4	4	3
Матеріально-технічна база	3	3	3
Термін реалізації ідеї	4	4	4
Супровідна документація	3	3	3
Сума	42	40	41
Середньоарифметична сума балів	(42+40+41) / 3 = 41		

За даними таблиці 4.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 4.3.

Таблиця 4.3 - Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	**Рівень комерційного потенціалу розробки**
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є високим, що досягається за рахунок того, що програмний продукт відрізняється від існуючих. Завдяки створеній підсистемі з’являється можливість ефективного безперервного планування та прийняття рішень у режимі реального часу. Особливістю підсистеми є те, що вона працює за власними математичними моделями, завдяки яким студент вказує запланований час вивчення обраної теми, а підсистема після її вивчення підраховує реальний витрачений час та автоматично змінює запланований час вивчення наступної теми в залежності від успішності та витраченого часу студента.

4.2 Прогнозування витрат на виконання науково-дослідної (дослідно-конструкторської) роботи

Основна заробітна плата розробників, яка розраховується за формулою:

$$З_о = \frac{M}{T_р} \cdot t, \quad (4.1)$$

де М — місячний посадовий оклад конкретного розробника (дослідника), грн.;

Тр — число робочих днів в місяці, 23 днів;

t — число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 4.4.

Таблиця 4.4 – Основна заробітна плата розробників

Найменування посади	**Місячний посадовий оклад, грн.**	**Оплата за робочий день, грн.**	**Число днів роботи**	**Витрати на заробітну плату, грн.**
Керівник проекту	40000	1739,13	36	62608,696
Програміст	35000	1521,74	36	54782,609
Всього				117391,30

Додаткова заробітна плата прийнято розраховувати як 13% від основної заробітної плати розробників та робітників:

$$З_д = З_о \cdot 13\% / 100\%, \quad (4.2)$$

$$З_д = (117391,30 \cdot 13\% / 100\%) = 15260,87 \text{ (грн)}.$$

Згідно діючого законодавства нарахування на заробітну плату складають 22% від суми основної та додаткової заробітної плати:

$$Н_з = (З_о + З_д) \cdot 22\ \% / 100\%, \quad (4.3)$$

$$Н_з = (117391{,}30 + 15260{,}87) \cdot 22\ \% / 100\ \% = 29183{,}48\ (грн).$$

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді амортизація обладнання, що використовувалась для розробки розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12}, \quad (4.4)$$

$$А = (48128 / 2) \cdot (1{,}57 / 12) = 3438{,}78\ (грн),$$

де Ц — балансова вартість обладнання, грн;

Т — термін корисного використання обладнання згідно податкового законодавства, років;

$t_{вик}$ — термін використання під час розробки, місяців.

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 4.5.

Так як вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів менше 20000 грн (операційна система «Microsoft Windows 10 Pro» – 7899 грн., інтегроване середовище розробки «Android Studio» – безкоштовно), то даний нематеріальний актив не амортизується, а його вартість включається у вартість розробки повністю, $В_{нем.ак.} = 7899$ грн.

Також потрібно включити у вартість нематеріальних ресурсів і вартість підписки на такі нематеріальний актив як база даних «Firebase» — 6702,4грн. в місяць; програмний продукт «Microsoft 365» — 158,25 грн./міс; крамниця застосунків «Google Play Pass» — для України 37,5 грн. в місяць. Активи використовувалися 1,57 місяця.

Таблиця 4.5 – Амортизаційні відрахування матеріальних і нематеріальних ресурсів для розробників

Найменування обладнання	**Балансова вартість, грн.**	**Строк корисного використан- ня, років**	**Термін використання обладнання, місяців**	**Амортиза-ційні відрахуван-ня, грн.**
Комп'ютер та комп'ютерна периферія (Intel Core i3-10100F 3.6 GHz / 6 MB, MSI B450-A Pro Max, Chieftec CTG-650C, Kingston NV1 1TB NVMe M.2 2280 PCIe 0x4, HyperX DDR4-3200 16384MB PC4-25600, Gigabyte Geforce Gtx Pci-Ex 1060 3Gb, DeepCool Gammaxx 400XT, Tecware Nexus Evo, MSI Optix G241, HyperX Alloy FPS Pro, HyperX Pulsefire Surge USB, HP Laser 107a)	48128	2	1,57	3438,78
Офісне обладнання (меблі)	20000	4	1,57	652,174
Приміщення	1000000	20	1,57	6521,739
Всього				10312,70

$$В_{нем.ак.} = 10490,713 + 247,70 + 58,70 + 7899 = 18696,10 \text{ грн.}$$

Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (4.5)$$

$$В_е = 0,9 \cdot 0,5 \cdot 8 \cdot 36 \cdot 6,2 = 803,52 \text{ (грн)},$$

де В — вартість 1 кВт-години електроенергії для 1 класу підприємства, В = 6,2 грн./кВт;

П — встановлена потужність обладнання, кВт. П = 0,5 кВт;

Ф — фактична кількість годин роботи обладнання, годин;

$К_п$ — коефіцієнт використання потужності, $К_п$ = 0,9.

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.6)$$

$$I_в = 117391{,}30 * 75\% / 100\% = 88043{,}48 \text{ (грн)},$$

де $Н_{ів}$ — норма нарахування за статтею «Інші витрати».

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$Н_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.7)$$

$$Н_{нзв} = 117391{,}30 * 115\ \% / 100\ \% = 135000 \text{ (грн)},$$

де $Н_{нзв}$ — норма нарахування за статтею «Накладні (загальновиробничі) витрати».

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються *ЗВ*, визначається за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \quad (4.8)$$

$$ЗВ = 414691{,}45 / 0{,}5 = 829383 \text{ грн},$$

де η — коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$$B_{заг} = 117391,30 + 15260,87 + 29183,48 + 10312,70 + 18696,10 + 803,52 + \\ + 88043,48 + 135000 = 414691,45 \text{ грн.}$$

4.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

Майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (4.9)$$

$$\Delta П1 = (0 * 50 + (1000 + 50) * 18000) * 0,8333 * 0,3) * (1 - 0,18) = 3689999,852 \text{ грн},$$

$$\Delta П2 = (0 * 50 + (1000 + 50) * (18000 + 20000) * 0,8333 * 0,3) * (1 - 0,18) = \\ = 8179499,673 \text{ грн},$$

$$\Delta П3 = (0 * 50 + (1000 + 50) * (18000 + 20000 + 22000) * 0,8333 * 0,3) * (1 - 0,18) = \\ = 12914999,483 \text{ грн},$$

де $\pm\Delta Ц_о$ — зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N — кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

$Ц_о$ — основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_о = Ц_б \pm \Delta Ц_о$;

$Цб$ — вартість програмного продукту у році до впровадження результатів розробки;

ΔN — збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ — коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$.

p — коефіцієнт, який враховує рентабельність продукту;

ϑ — ставка податку на прибуток, у 2022 році ϑ =18%.

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 24784499,01 грн.

Розраховуємо приведену вартість збільшення всіх чистих прибутків ПП:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.10)$$

ПП = (3689999,852 / (1 + 0,1)1) + (8179499,673 / (1 + 0,1)2) + (12914999,483 / / (1 + 0,1)3) = 3354545,32 + 6759917,085 + 9703230,266 = 19817692,67 грн,

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

Т — період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0{,}05 \ldots 0{,}15$;

t — період часу (в роках).

Далі розраховують величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} * ЗВ, \quad (4.11)$$

PV = 2 * 829383 = 1658765,80 грн,

де $k_{інв}$ — коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію.

ЗВ — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

Тоді абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV, \tag{4.12}$$

$$E_{абс} = 19817692{,}67 - 1658765{,}80 = 18158926{,}87 \text{ грн},$$

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$.

Для цього використаємо формулу:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.13}$$

$$E_в = \sqrt[3]{(1 + 18158926{,}87/1658765{,}80} - 1 = 1{,}286,$$

де $T_ж$ — життєвий цикл наукової розробки, роки.

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \tag{4.14}$$

$$\tau_{\min} = 0{,}14 + 0{,}05 = 0{,}19,$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d = (0,09…0,14);

f — показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}, \qquad (4.15)$$

$$T_{ок} = 1 / 1{,}286 = 0{,}78 \text{ р.}$$

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 0,78 роки, то фінансування даної наукової розробки є доцільним.

ВИСНОВКИ

У магістерській дипломній роботі було створено підсистему коригування термінів вивчення обраних дисциплін системи дистанційного навчання.

Підсистему розроблено для ефективного планування часу вивчення навчальних дисциплін та прийняття рішень у режимі реального часу.

Після детального аналізу предметної області щодо особливостей організації планування, встановлено, що планування є універсальним інструментом організації діяльності у різних галузях, зокрема: освіта, економіка, спорт. На основі проведених досліджень доведено ефективність застосування планування під час визначення цілей та методів їх досягнення на основі комплексу завдань й робіт, а також визначення різних продуктивних методів, ресурсів та способів, які необхідні, щоб виконати обрані завдання, та встановити їх взаємозв'язки. Для створення програмного забезпечення використано технології діаграми Ґанта, що використовуються для вивчення дисциплін.

У ході аналізу аналогів, виявлено програмні забезпечення із подібними функціоналом та призначенням, проте підсистема, яка планується до розробки, має значні функціональні переваги. Обґрунтовано доцільність розробки підсистеми коригування термінів вивчення обраних дисциплін системи дистанційного навчання, оскільки вона дозволяє вказувати та редагувати терміни одночасно та безкоштовно у порівнянні з іншими аналогами.

Було обґрунтовано та розроблено модель врахування змін у термінах вивчення дисциплін та модель навчання. Після детального аналізу обрано найкращі варіанти для кожної із розглянутих моделей, що дозволить використовувати підсистему коригування термінів вивчення обраних дисциплін системи дистанційного навчання з максимальною ефективністю для студента.

Проведено проектування підсистеми коригування термінів вивчення обраних дисциплін системи дистанційного навчання. Здійснено реалізацію алгоритму роботи підсистеми та розглянуто структуру програми коригування термінів

вивчення обраних дисциплін, а також було побудовано UML-діаграму діяльності комп'ютерної програми. Для демонстрації взаємодії сервера та клієнта описано та розглянуто структуру класів підсистеми. Було побудовано UML-діаграму класів програми коригування термінів вивчення обраних дисциплін.

Було обґрунтовано вибір програмного середовища Android Studio, мови програмування Java та серверу бази даних Firebase.

Проведено проектування підсистеми коригування термінів вивчення обраних дисциплін системи дистанційного навчання. Розроблено архітектуру, базу даних та інтерфейс підсистеми коригування термінів вивчення обраних дисциплін системи дистанційного навчання.

Виконано тестування роботи підсистеми та проаналізовано результати працездатності підсистеми. Створено інструкцію користувача для користування розробленою підсистемою коригування термінів вивчення обраних дисциплін системи дистанційного навчання.

Економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 829383 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 0,78 роки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Програмне забезпечення для врахування змін у термінах вивчення дисциплін за розкладом на основі діаграми Ґанта [Текст] / В. О. Бучинський, А. В. Снігур. // L Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії (2021) : Тез. доп. — Вінниця, 2021. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2021/paper/view/12067 (дата звернення 15.11.22).

2. Васильців Т. Г., Качмарик Я. Д., Блонська В. І., Лупак Р. Л. Бізнес-планування: посібник. Київ : Знання, 2013. 173 с.

3. Рассел Д. Диаграмма Ганта. Вид. : VSD, 2017. С. 591

4. Эрджиес К. Распределенные системы реального времени. Теория и практика. Вид. : ДМК Пресс, 2020. С. 382

5. Маккеон Ґ. Есенціалізм. Мистецтво визначати пріоритети. Вид. : Наш формат, 2021. С. 224

6. Федотенко М. Разработка мобильных приложений. Первые шаги. Вид. : Бином. Лаборатория знаний, 2016. С. 335

7. Кноблаух Й., Вёльте Х. Управление временем. 2-ге вид. : Омега-Л, 2006. С. 144

8. Маркова Є. С. Інформаційні технології навчання: посібник. Запоріжжя : Просвіта, 2012. 118 с.

9. Гриценчук О. О., Коневщинська О. Е. Інформаційні та комунікаційні технології навчання в системі загальної середньої освіти зарубіжних країн : посібник. Київ : Педагогічна думка, 2012. 176 с.

10. Розробка мобільних додатків від А до Я [Електронний ресурс]. — 2021. — Режим доступу: https://dan-it.com.ua/uk/rozrobka-mobilnih-dodatkiv-vid-a-do-ja-povnij-gajd/ (дата звернення 05.11.22).

11. Бурнет Э. Привет, Android! Разработка мобильных приложений. Вид. : Ветер, 2016. С. 256

12. Кнут Д. Искусство программирования. Основные алгоритмы. Вид. : Вильямс, 2005. С. 720

13. Дарвин Я. Android. Сборник рецептов. Задачи и решения для разработчиков приложений. Вид. : Вильямс, 2016. С. 768

14. Колисниченко Д. Программирование для Android. Самоучитель. 3-тє вид. : БХВ, 2020. С. 288

15. Учебник по Firebase [Електронний ресурс]. — 2019. — Режим доступу: https://coderlessons.com/tutorials/veb-razrabotka/izuchite-firebase/uchebnik-po-firebase (дата звернення 07.11.22).

16. Архітектура «клієнт-сервер» [Електронний ресурс]. — 2018. — Режим доступу: https://it.wikireading.ru/32781 (дата звернення 13.11.22).

17. Побудова SIFT дескрипторів і задача зіставлення зображень [Електронний ресурс]. — 2020. — Режим доступу: https://habr.com/post/106302/ (дата звернення 14.11.22).

18. Принципи та форми організації багатокористувацьких інформаційних систем [Електронний ресурс]. — 2015. — Режим доступу: http://um.co.ua/11/11-7/11-73087.html (дата звернення 16.11.22).

19. Software Development Articles [Електронний ресурс]. — 2021. — Режим доступу: http://www.softdevarticles.com/modules/weblinks/viewcat.php?cid=21 (дата звернення 18.11.22).

20. Matthew West [Електронний ресурс]. — 2017. — Режим доступу: http://www.matthew-west.org.uk/documents/princ03.pdf (дата звернення 21.11.22).

21. Дейтел Х. Технологии программирования на Java 2. Графика, JavaBeans, интерфейс пользователя. Вид. : Бином-пресс, 2013. С. 210

22. Котляров В., Коликова Т. Основы тестирования программного обеспечения. Вид. : Бином. Лаборатория знаний, 2016. С. 321

ДОДАТОК А

Технічне завдання

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
проф., д.т.н.. Азаров О.Д..

" " 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

"Підсистема коригування термінів вивчення обраних дисциплін системи дистанційного навчання"

08-23.МКР.001.00.000.ТЗ

Науковий керівник: доцент к.т.н.
_____ Снігур А.В.

Студент групи 1КІ-21м
_____ Бучинський В.О.

ВНТУ 2022

1 Підставою для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Підставою для виконання магістерської кваліфікаційної роботи є процес створення оптимальної структури навчального графіка, забезпечення безперервного планування та коригування термінів вивчення обраних дисциплін системи дистанційного навчання

1.2 Наказ про затвердження теми МКР.

2 Мета МКР і призначення розробки

2.1 Мета роботи — розробка підсистеми для ефективного планування часу вивчення навчальних дисциплін;

2.2 Призначення розробки — виконання магістерської дипломної роботи та створення підсистеми для користування студентами.

3 Вихідні дані для виконання МКР

3.1 Проведення аналізу існуючих принципів програмних засобів та навчальних систем, які використовують діаграму Ґанта для планування;

3.2 Розробка структури підсистеми коригування термінів вивчення обраних дисциплін системи дистанційного навчання;

3.3 На основі структурних схем здійснено моделювання та проектування підсистеми коригування термінів вивчення обраних дисциплін системи дистанційного навчання;

3.4 Виконання розрахунків для доведення доцільності нової розробки з економічної точки зору;

4 Вимоги до виконання МКР

Головна вимога — використати, як основний, метод врахування змін у термінах вивчення дисциплін та дотримуватись методу навчання.

5 Етапи МКР та очікувані результати

Таблиця А.1 — Етапи МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз сучасного стану досліджень	10.09.2022	19.09.2022	Аналітичний огляд літературних джерел, задачі досліджень,
2	Аналітичний огляд програмних засобів та навчальних систем, які використовують діаграму Ґанта для планування	20.09.2021	29.09.2022	Розділ 1
3	Моделювання та проектування підсистеми коригування термінів вивчення обраних дисциплін системи дистанційного навчання	02.10.2022	12.10.2022	Розділ 2
4	Проектування та опис роботи підсистеми коригування термінів вивчення обраних дисциплін системи дистанційного навчання	13.10.2022	22.10.2022	Розділ 3
5	Економічна частина	23.10.2022	30.10.2022	Розділ 4
6	Практична реалізація	31.10.2022	13.11.2022	Програмний продукт
7	Апробація та впровадження результатів дослідження	14.11.2022	20.11.2022	Тези доповідей
8	Опублікування результатів досліджень	21.11.2022	27.11.2022	Патент
9	Оформлення пояснювальної записки, презентації	28.11.2022	02.12.2022	Пояснювальна записка, матеріал і презентація
10	Підготовка супроводжуючих документів, їх підписування, проходження нормоконтролю та тесту на плагіат	03.12.22	10.12.2022	Оформлені документи

6 Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до

МКР українською та іноземною мовами, довідка про відповідність оформлення МКР діючим вимогам.

7 Порядок контролю виконання та захисту МКР

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

8 Вимоги до оформлювання та порядок виконання МКР

8.1 При оформлювання МКР використовуються:

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

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— міждержавний ГОСТ 2.104-2006 «Едина система конструкторської документації. Основні написи»;

— Методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія». Кафедра обчислювальної техніки ВНТУ 2022;

— документами на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21».

Технічне завдання до виконання отримав Бучинський В. О.

ДОДАТОК Б

Лістинг «MainActivity»

```
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.Button;
import android.widget.RelativeLayout;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import com.example.volodymyr4kbakalavrwork.Models.Userupdate;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.material.snackbar.Snackbar;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.firestore.auth.User;
import com.rengwuxian.materialedittext.MaterialEditText;
import java.util.Objects;
public class MainActivity extends AppCompatActivity {
Button button_sign_in, button_register_now;
FirebaseAuth auth;
FirebaseDatabase db;
```

```
DatabaseReference users;
RelativeLayout root;
        @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(com.example.volodymyr4kbakalavrwork.R.layout.activity_main);
button_sign_in = findViewById(R.id.button_sign_in);
button_register_now = findViewById(R.id.button_register_now);
root = findViewById(R.id.root_element);
        auth = FirebaseAuth.getInstance();
db = FirebaseDatabase.getInstance();
users = db.getReference("Users");
button_register_now.setOnClickListener(v -> showRegisterWindow());
button_sign_in.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
showSignInWindow();
         }
        });
        }
private void showSignInWindow() {
AlertDialog.Builder dialog = new AlertDialog.Builder(this);
dialog.setTitle("Увійти");
        dialog.setMessage("Введіть дані для входу в обліковий запис");
 LayoutInflater inflater = LayoutInflater.from(this);
View sign_in_window = inflater.inflate(R.layout.sign_in_window, null);
        dialog.setView(sign_in_window);
        final MaterialEditText email = sign_in_window.findViewById(R.id.email);
final MaterialEditText password = sign_in_window.findViewById(R.id.password);
```

```
dialog.setNegativeButton("Повернутись", (dialogInterface, which) ->
dialogInterface.dismiss());
dialog.setPositiveButton("Увійти", (dialogInterface, which) -> {
        if(TextUtils.isEmpty(email.getText().toString())) {
Snackbar.make(root, "Ви не ввели електронну пошту користувача",
Snackbar.LENGTH_SHORT).show();
return;     }
if(password.getText().toString().length() < 4 ) {
Snackbar.make(root, "Ви не ввели недостатню кількість символів для паролю
користувача", Snackbar.LENGTH_SHORT).show();
 return;        }
auth.signInWithEmailAndPassword(email.getText().toString(),
password.getText().toString())
addOnSuccessListener(new OnSuccessListener<AuthResult>() {
@Override
public void onSuccess(AuthResult authResult) {
startActivity(new Intent(MainActivity.this, DaysActivity.class));
        finish();
        }
})
        .addOnFailureListener(new OnFailureListener() {
@Override
public void onFailure(@NonNull Exception e) {
        Snackbar.make(root, "Помилка при авторизації." + e.getMessage(),
Snackbar.LENGTH_SHORT ).show();
        }
        });
        });
dialog.show();
```

```
        }
private void showRegisterWindow() {
AlertDialog.Builder dialog = new AlertDialog.Builder(this);
dialog.setTitle("Зареєструватись");
dialog.setMessage("Введіть дані для реєстрації");
LayoutInflater register = LayoutInflater.from(this);
View register_window = register.inflate(R.layout.register_window, null);
dialog.setView(register_window);
final MaterialEditText login = register_window.findViewById(R.id.username);
final MaterialEditText email = register_window.findViewById(R.id.email);
final MaterialEditText password = register_window.findViewById(R.id.password);
final MaterialEditText passaccept =
register_window.findViewById(R.id.password_accept);
dialog.setNegativeButton("Повернутись", (dialogInterface, which) ->
dialogInterface.dismiss());
dialog.setPositiveButton("Підтвердити", (dialogInterface, which) -> {
if(TextUtils.isEmpty(Objects.requireNonNull(login.getText()).toString())) {
Snackbar.make(root, "Ви не ввели ім'я користувача",
Snackbar.LENGTH_SHORT).show();
return;                                          }
if(TextUtils.isEmpty(email.getText().toString())) {
Snackbar.make(root, "Ви не ввели електронну пошту користувача",
Snackbar.LENGTH_SHORT).show();
return;                          }
if(password.getText().toString().length() < 4 ) {
Snackbar.make(root, "Ви не ввели недостатню кількість символів для паролю
користувача", Snackbar.LENGTH_SHORT).show();
return;                            }
if(passaccept.getText().toString().length() < 4 ) {
```

```
Snackbar.make(root, "Ви не ввели недостатню кількість символів для паролю користувача", Snackbar.LENGTH_SHORT).show();
return;                         }
if(!password.getText().toString().equals(passaccept.getText().toString())) {
Snackbar.make(root, "Паролі користувача не співпадають",
Snackbar.LENGTH_SHORT).show();
return;                                                 }
auth.createUserWithEmailAndPassword(email.getText().toString(),
password.getText().toString())
.addOnSuccessListener(authResult -> {
        Userupdate user = new Userupdate();
user.setLogin(login.getText().toString());
user.setEmail(email.getText().toString());
user.setPassword(password.getText().toString());
user.setPassaccept(passaccept.getText().toString());
        users.child(FirebaseAuth.getInstance().getCurrentUser().getUid())
        .setValue(user)
        .addOnSuccessListener(aVoid -> Snackbar.make(root, "Реєстрація завершена",
Snackbar.LENGTH_SHORT).show());
}).addOnFailureListener(new OnFailureListener() {
@Override
public void onFailure(@NonNull Exception e) {
Snackbar.make(root, "Помилка при авторизації." + e.getMessage(),
Snackbar.LENGTH_LONG).show();
        }
        });
         });
        dialog.show();    }
        }
```

ДОДАТОК В

Лістинг «SubjectActivity»

```
public class SubjectActivity extends AppCompatActivity {
Button btnSubject1, btnSubject2, btnSubject3, btnSubject4, @Override
 protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_day);
btnSubject1 = findViewById(R.id.subject_one);
btnSubject2 = findViewById(R.id.subject_two);
btnSubject3 = findViewById(R.id.subject_three);
btnSubject4 = findViewById(R.id.subject_four);
btnSubject5 = findViewById(R.id.subject_five);
btnProfile.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
showProfileWindow();
 }
);
btnReturn.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
startActivity(new Intent(SubjectActivity.this, DayActivity.class));
finish();
}
);
btnSubject1.setOnClickListener(new View.OnClickListener() {
@Override
 public void onClick(View v) {
```

```
startActivity(new Intent(SubjectActivity.this, SubjectOneActivity.class));
finish();
}     }
);
btnSubject2.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
startActivity(new Intent(SubjectActivity.this, SubjectTwoActivity.class));
finish();        }
);
btnSubject3.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
startActivity(new Intent(SubjectActivity.this, SubjectThreeActivity.class));
finish();       }
}
);
btnSubject4.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
startActivity(new Intent(SubjectActivity.this, SubjectFourActivity.class));
finish();    }   }
);
btnSubject5.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
startActivity(new Intent(SubjectActivity.this, SubjectFiveActivity.class));
finish();         }
);   }
```

ДОДАТОК Г

Лістинг «SubjectTopicActivity»

```
public class SubjectTopicActivity extends AppCompatActivity {
Button btnTopic1, btnTopic2, btnTopic3, btnTopic4, btnTopic5, btnGanttDiagram;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_economy);
btnTopic1 = findViewById(R.id.subject_topic_one);
btnTopic2 = findViewById(R.id.subject_topic_two);
btnTopic3 = findViewById(R.id.subject_topic_three);
btnTopic4 = findViewById(R.id.subject_topic_four);
btnTopic5 = findViewById(R.id.subject_topic_five);
btnGanttDiagram = findViewById(R.id.subject_gantt_diagram);
btnProfile.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
showProfileWindow();
 }
});
btnReturn.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
startActivity(new Intent(SubjectTopicActivity.this, SubjectActivity.class));
finish();
 }
});
btnTopic1.setOnClickListener(new View.OnClickListener() {
```

```
@Override
public void onClick(View v) {
startActivity(new Intent(SubjectTopicActivity.this,
SubjectTopicOneActivity.class));
finish();
}
});
btnTopic2.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) { startActivity(new Intent(SubjectTopicActivity.this,
SubjectTopicTwoActivity.class));
finish();
}
});
btnTopic3.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
startActivity(new Intent(SubjectTopicActivity.this,
SubjectTopicThreeActivity.class));
finish();
}
});
btnTopic4.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
startActivity(new Intent(SubjectTopicActivity.this,
SubjectTopicFourActivity.class));
finish();
}
```

```
        });
        btnTopic5.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
        startActivity(new Intent(SubjectTopicActivity.this,
SubjectTopicFiveActivity.class));
        finish();
         }
        });
        btnGanttDiagram.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
        startActivity(new Intent(SubjectTopicActivity.this, SubjectGanttDiagram.class));
        finish();
        }
        })
        }
```

ДОДАТОК Д

Лістинг «GanttDiagramActivity»

```
public class GanttDiagramActivity extends AppCompatActivity {
Button btnProfile, btnReturn;
int plantimetopicone, plantimetopictwo, plantimetopicthree, plantimetopicfour, plantimetopicfive, plantimetopics;
int RealTimeTopicOne, int RealTimeTopicTwo, int RealTimeTopicThree, int RealTimeTopicFour, int RealTimeTopicFive;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_gantt_diagram_building);
btnProfile = findViewById(R.id.days_window_profile);
btnReturn = findViewById(R.id.days_window_cancel);
btnProfile.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
showProfileWindow();
}
 });
btnReturn.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
startActivity(new Intent(GanttDiagramActivity.this, GanttDiagramActivity.class));
finish();
}
});
plantimetopics()
```

```
}
public void plantimetopics() {
plantimetopicone = $('#plantime_topicone').val();
plantimetopictwo = $('#plantime_topictwo').val();
plantimetopicthree = $('#plantime_topicthree').val();
plantimetopicfour = $('#plantime_topicfour').val();
plantimetopicfive = $('#plantime_topicfive').val();
plantimetopics = plantimetopicone + plantimetopictwo + plantimetopicthree +
plantimetopicfour + plantimetopicfive;
Bundle extras1 = getIntent().getExtras();
if (extras1 != null) {
 RealTimeTopicOne = FirebaseDatabase_query("SELECT * FROM realtime_topics
WHERE realtimetopicone = "'.$realtime_topicone.'"");
}
Bundle extras2 = getIntent().getExtras();
if (extras2 != null) {
RealTimeTopicTwo = FirebaseDatabase_query("SELECT * FROM realtime_topics
WHERE realtimetopictwo = "'.$realtime_topictwo.'"");
}
Bundle extras3 = getIntent().getExtras();
if (extras3 != null) {
RealTimeTopicThree = FirebaseDatabase_query("SELECT * FROM
realtime_topics WHERE realtimetopicthree = "'.$realtime_topicthree.'"");
 }
Bundle extras4 = getIntent().getExtras();
if (extras4 != null) {
RealTimeTopicFour = FirebaseDatabase_query("SELECT * FROM realtime_topics
WHERE realtimetopicfour = "'.$realtime_topicfour.'"");
}
```

```
        Bundle extras5 = getIntent().getExtras();
        if (extras5 != null) {
        RealTimeTopicFour = FirebaseDatabase_query("SELECT * FROM realtime_topics
WHERE realtimetopicfive = '".$realtime_topicfive."'");
        }
        if (plantimetopicone > RealTimeTopicOne) {
        plantimetopictwo = plantimetopictwo + (plantimetopicone - RealTimeTopicOne);}
        if (plantimetopictwo > RealTimeTopicTwo) {
        plantimetopicthree = plantimetopicthree + (plantimetopictwo -
RealTimeTopicTwo);
        }
        if (plantimetopicthree > RealTimeTopicThree) {
        plantimetopicfour = plantimetopicfour + (plantimetopicthree -
RealTimeTopicThree);
        }
        if (plantimetopicfour > RealTimeTopicFour) {
        plantimetopicfive = plantimetopicfive + (plantimetopicfour - RealTimeTopicFour);}
        if (plantimetopicone < RealTimeTopicOne) {
        plantimetopictwo = plantimetopictwo - (RealTimeTopicOne - plantimetopicone); }
        if (plantimetopictwo < RealTimeTopicTwo) {
        plantimetopicthree = plantimetopicthree - (RealTimeTopicTwo - plantimetopictwo);
}
        if (plantimetopicthree < RealTimeTopicThree) {
        plantimetopicfour = plantimetopicfour - (RealTimeTopicThree -
plantimetopicthree);
        }
        if (plantimetopicfour < RealTimeTopicFour) {
        plantimetopicfive = plantimetopicfive – (RealTimeTopicFour - plantimetopicfour);}
}
```

ДОДАТОК Е

UML-діаграма діяльності підсистеми

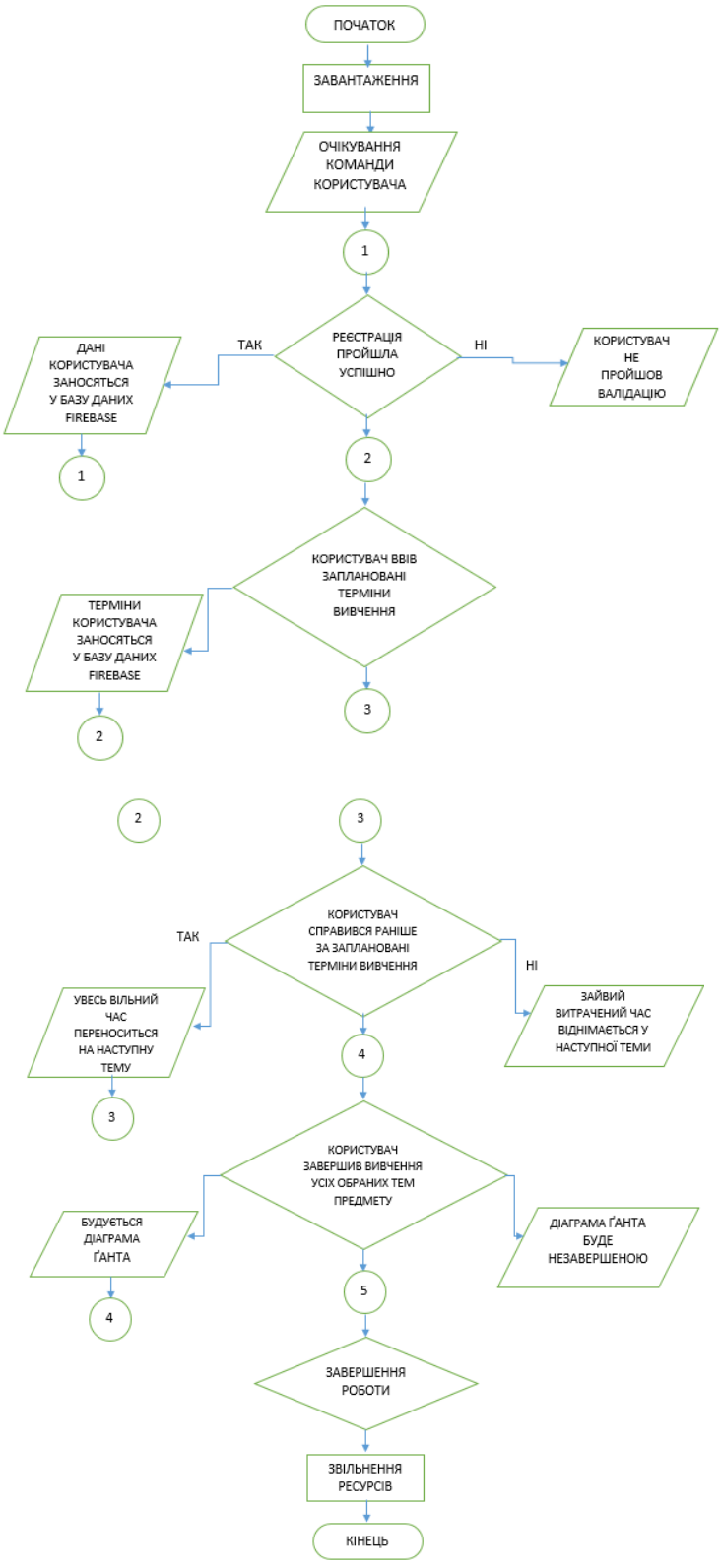

Рисунок Е.1 — UML-діаграма діяльності комп’ютерної програми

ДОДАТОК Ж

Протокол перевірки навчальної (кваліфікаційної) роботи

Назва роботи: Підсистема коригування термінів вивчення обраних дисциплін системи дистанційного навчання

Тип роботи: магістерська кваліфікаційна робота

(БДР, МКР)

Підрозділ кафедра обчислювальної техніки

(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 97% Схожість 3%

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 _____ Бучинський В.О.

(підпис) (прізвище, ініціали)

Керівник роботи _____ Снігур А. В.

(підпис) (прізвище, ініціали)