МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ САДІВНИЦТВА

Факультет економіки і підприємництва

Кафедра інформаційних технологій

Допущено до захисту
"_____" _____ 2023 р.
Зав. кафедри, к.т.н., доцент
_____ **Ліщук Р.І.**

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

РОЗРОБКА АЛГОРИТМУ ДЛЯ ОПТИМАЛЬНОГО РОЗПОДІЛУ ЧАСУ ПРИ ТЕСТУВАННІ СТУДЕНТІВ

Виконав – студент 2 курсу, групи 21к-кн
освітньої програми "Комп'ютерні науки"
спеціальності 122 Комп'ютерні науки
Гапон Дмитро Сергійович

Науковий керівник – к.е.н., доцент
Концеба Сергій Михайлович

Умань – 2023

ЗМІСТ

ВСТУП 4

РОЗДІЛ 1. ІНФОРМАЦІЙНИЙ ОГЛЯД 5

1.1 Поняття СУБД. Моделі баз даних. Реляційна модель 5

1.2 Опис СУБД MariaDB 6

1.3 Опис основних мов програмування 7

1.4 Поняття алгоритму. Нейронних мереж. IEI-технологія 8

1.5 Постановка задачі 9

РОЗДІЛ 2. ВИБІР МЕТОДІВ ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ 11

2.1 Проектування бази даних 11

2.2 Вибір середовища розробки 13

2.3 Визначення команд управління системою 14

2.4 Визначення основних методів програмування додатку 16

2.5 Адаптивний метод підбору часу 17

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ 20

3.1 Створення бази даних 20

3.2 Приклад типових запитів до бази даних 22

3.3 Програмна реалізація додатку 23

3.4 Приклад тестування програми та результату роботи з програми 30

ВИСНОВКИ 39

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 40

ДОДАТКИ 42

ВСТУП

Платформа Телеграм[1] - являється некомерційним проектом що має також відкритий вихідний код. Телеграм кросплатформений, тому його можна відкрити на декількох різних девайсах. Особливістю месенджеру є наявність Bot API[3], що може реалізовувати логіку для написання ботів.

Боти[2] - спеціальні акаунти в Telegram, створені для того, щоб автоматично обробляти та відправляти повідомлення. Звичайні користувачі маєть можливість взаємодіяти з аккаунтами ботів за допомогою миттєвих повідомлень, що відправляються через групові або звичайні чати. Логіка бота контролюється за допомогою HTTPS запитів до нашого API для пошукових роботів. Особливість роботи бота - принцип запит-відповідь, бот реагує на надіслане йому повідомлення, обробляє його та надсилає йому відповідь.

В ході роботи буде реалізовано адаптивний алгоритм розрахунку часових проміжків для автоматизованої системи тестувань студентів на основі інтерфейсу бота. Алгоритм має адаптуватися до швидкості проходження тесту різними студентами, вірності виконання завдання.

Додаток буде використовуватися для проведення тестування студентів на підготовчих курсах та у навчальному процесі. За допомогою алгоритму студенти будуть виконувати завдання у оптимальний час для уникання надмірного споживання часу.

РОЗДІЛ 1. ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1. Поняття СУБД. Моделі баз даних. Реляційна модель

Система керування базами даних[20] (СУБД) [4] — набір взаємопов'язаних даних, які називаються базою даних, а також набір програм для можливого доступу до цих даних. Надає можливості оновлення, створення, пошуку і збереження даних у базах даних із контролем доступу до збереженої інформації.

Можливості СУБД:

1. Є можливість створення БД

2. Має надавали доступ до операцій вставки, видалення, оновлення та читання інформації з БД

3. Є можливість надавати користувачам контрольований доступ до БД

Модель бази даних[5] - тип моделі даних, яка визначає логічну структуру бази даних і принципово визначає, яким чином дані можуть бути збережені, організовані і оброблені. Найбільш популярним прикладом моделі бази даних є реляційна модель, яка використовує табличний формат.

У світі технологій баз даних існує два основних напрямки[6]: SQL і NoSQL, реляційні та нереляційні бази даних. Відмінності між ними полягають в тому, як вони спроектовані, які типи даних підтримують, як зберігають інформацію.

1. Реляційні БД[7] зберігають структуровані дані, які зазвичай представляють об'єкти реального світу. Це можуть бути, скажімо, короткі відомості про людину, або про товари у кошику в магазині, що згруповані в таблицях, формат товарів, в той час, заданий уже на етапі проектування сховища.

Причини, які можуть послужити приводом для вибору SQL-бази:

1. Необхідність виконувати вимоги ACID, а саме - атомарність, несуперечливість, ізольованість та довговічність. Це допомагає зменшити

ймовірність непередбачуваної поведінки системи і забезпечити цілісність даних у базі.

2. Основна структура даних не буде змінюватись, а дані можливо структурувати.

SQL-бази можуть бути застосовані в проетах такого типу:

1. Вимоги до даних можливо знати заздалегідь.

2. Технологія повинна бути надійна і добре рекомендована. Щоб можна було розраховувати на досвіт освічених розробників, а недоліки були відомі попередньо.

3. Цілісність даних пріорітетна.

1.2. Опис СУБД MariaDB

MariaDB[20] фактично - це відгалуження від СУБД MySQL, що розробляється спільнотою під ліцензією GNU GPL. Розробка та підтримка сайта MariaDB здійснює компанія MariaDB Corporation Ab і фонд MariaDB Foundation. Поштовхом до створення стала необхідність забезпечення вільного статусу СУБД, на противагу політиці ліцензування MySQL компанією Oracle. Система ліцензування MariaDB зобов'язує учасників, бажаючих додати свій код в основну гілку СУБД, обмінюватися своїми авторськими правами з MariaDB Foundation для охорони ліцензії і можливості створювати критичні виправлення для MySQL.

Вона повністю сумісна з MySQL, і прекрасно підходить в якості заміни, тому що повністю відповідає як набір команд, так і API.

Особливості:

1. Система працює швидко.

2. Індикатори дадуть вам знати, як обробляється запит.

3. Розширювана архітектура і плагіни дозволяють налаштовувати інструмент відповідно до ваших потреб.

4. Шифрування є в мережі, сервері і на рівні додатку.

5. Рушій досить новий, тому поки немає ніяких гарантій подальших оновлень.

6. Як і в багатьох інших безкоштовних базах даних, вам доведеться платити за підтримку.

7. Ідеальна як альтернатива MySQL, якщо MySQL не влаштовує з якихось причин.

MariaDB являється відгалудженням СУБД Mysql, яка має в перевазі ті ж характеристики і особливості, але, ліцензію на Mysql викупила Oracle[8] і частково випускається згідно своїх ліцензій, натомість MariaDB – повністю безкоштовна СУБД з відкритим вихідним кодом. І, як нове відгалудження від Mysql [9], MariaDB [10] частіше випускає свої оновлення. MariaDB зараз стрімко розвивається і має більше можливостей. Ці можливості стосуються оптимізації, поліпшення роботи з пам'яттю, і багато іншого. Згідно попереднього, MariaDB [11] являється потрібною СУБД для даного проекту. Також, можна додати, що її подальшу реалізацію схеми на діалекті SQL можна з легкістю перенести на MySQL СУБД, адже синтаксис залишається тим же. Тому орієнтованою СУБД для розробки являється MariaDB [12]. Також MariaDB, як і MySQL успішно інтегруються з мовою програмування Java. Для MariaDB присутній JDBC driver, що робить операції з бд із програми безпечними та надійними.

1.3. Опис основних мов програмування

C++ – це мова програмування високого рівня[20], яка підтримує певні парадигми програмування: узагальненої, об'єктно-орієнтованої та процедурної. Мова як правило застосовується для системного програмування, написання драйверів, високопродуктивних програм, що будуть виконуватись на сервері або ж клієнтському комп’ютері.

C# – мова програмування, яка являється об'єктно-орієнтованою, з безпечною системою типізації, розроблена для платформи .NET. Перевагою

даної мови є те, що розмір вихідних програмних продуктів є відносно невеликим і не потребує додаткових бібліотек для запуску. Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, перевантаження операторів, вказівники на функції-члени класів, події, властивості, винятки, коментарі у форматі XML. Основуючись на своїх попередниках — мовах C++, Delphi і Smalltalk — C# виключає деякі потенційно нестабільні логічні моделі, як множинне спадкування класів.

Java [13] – також об'єктно-орієнтована мова програмування високого рівня, утім вона проектувалась саме у такій парадигмі від самого свого створення. В цьому вона вигідно відрізняється від C++, який, з міркувань сумісності, має багато успадкованих елементів процедурної мови C. Java [14] у великій мірі запозичила синтаксис із C і C++, наприклад, об'єктну модель було перейнято із C++, хоча і допрацьовано. Архітектурно була усунута можливість появи ряду конфліктних ситуацій та було спрощено сам процес розробки програм об'єктно-орієнтованих[15]. Ряд дій, які в C/C++ мають бути виконані програмістом, доручено в java віртуальній машині JVM [16].

1.4. Поняття алгоритму. Нейронних мереж. IEI-технологія

Алгоритм – це скінченна послідовність чітко визначених, реалізованих на комп'ютері інструкцій, як правило, для вирішення класу задач або для

обчислення. Алгоритми завжди однозначні і використовуються як специфікації для виконання розрахунків, обробки даних, автоматизованих міркувань та інших завдань. Нейронні мережі [18] (NN), - це обчислювальні системи, які нечітко натхненні біологічними нейронними мережами, що складають мозок тварин. Глибоке навчання є частиною більш широкого сімейства методів машинного навчання, заснованих на штучних нейронних мережах з репрезентативним навчанням. Глибоке навчання дозволяє навчати модель передбачати результат по набору вхідних даних. Головною одиницею являється нейрон, Нейрони згруповані в три різних типи шарів: вхідний шар;

прихований шар (шари); вихідний шар. випадку ми не матимемо стільки даних, яких система потребує для навчання. IEI-технологія[19] полягає у трансформації апріорного в загальному випадку нечіткого розбиття простору ознак до чіткого розбиття класів еквівалентності за допомогою ітераційною оптимізацією параметрів функціонування інформаційної системи. Основна ідея навчання за IEI-технологією полягає в послідовній нормалізації вхідного математичного опису ІС шляхом цілеспрямованої трансформації апріорних габаритів розкиду реалізацій образів з метою максимального їх захоплення контейнерами відповідних класів, що відбудовуються в радіальному базисі у процесі навчання.

1.5. Постановка задачі

Автоматизована система тестування реалізована на мові Java, інформаційна система спроектована в СУБД MariaDB. При проходженні тестування студенту дається певний час на виконання завдання (за замовчуванням – 5 хвилин), надалі, з проходженням тесту іншими студентами, система повинна адаптувати час, який надається для виконання завдання, адаптувати згідно до попередніх результатів, тобто за який час студенті відповідали або чи відповідали вірно.

Основна логіка описується як: запуском куратором тесту для групи, а потім проходження кожним студентом окремо тесту, через діалог з ботом. Паралельно куратор може переглядати результати проходження тестування групою. Розглянути вимоги можна детальніше:

1. Користувач в процесі реєстрації повинен вказати своє ім’я
2. Адмін створює групу для студентів і назначає кураторів для цієї групи.
3. У куратора є можливість створити тест для студентів, побачити список своїх створених тестів, починати тест для групи і побачити як студенти пройшли тест з балами.

4. Після початку тесту куратором, студент групи має отримати повідомлення про початок тесту, далі він відповідає на всі запитання і отримує від системи результат з оцінкою у процентному вигляді.

5. Студент має обмеження по часу для виконання завдання, при отриманні завдання він знає скільки часу відведено на завдання.

6. Якщо студент не встиг виконати завдання вчасно, завдання вважається незданим і студент переходить до іншого завдання.

РОЗДІЛ 2. ВИБІР МЕТОДІВ ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

2.1. Проектування бази даних

Логіка додатку базується на вводі даних від користувачів, обробкою їх системою паралельно записуючи проміжні дані до сховища та відповіді результату користувачу, деталі див на DFD діаграмі.

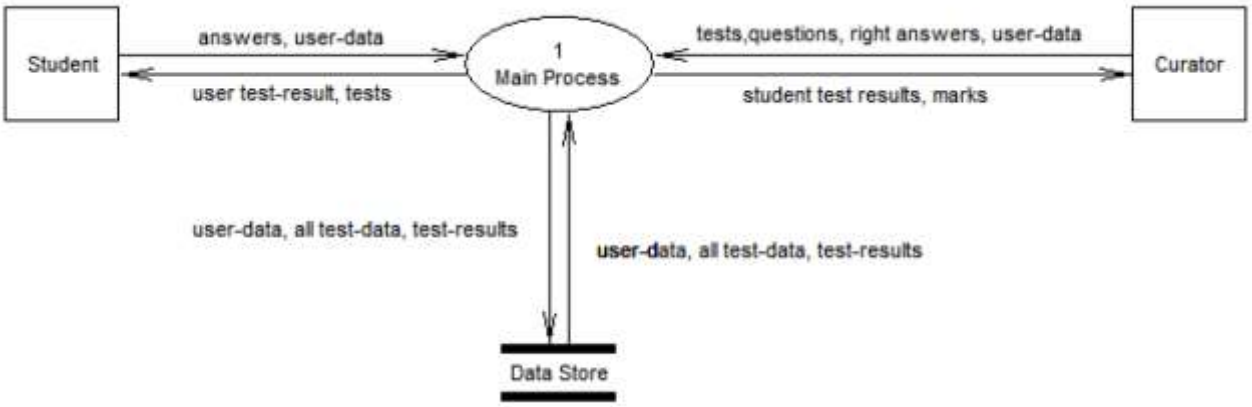


Рис. 2.1 DFD діаграма

Згідно до вимог, що вказані у пункті 1.5 мною була створена ERD діаграма рис 2.2,

user_id - унікальний ідентифікатор користувачів, що присвоюється платформою Телеграм. Основною таблицею в базі являється сутність users (користувачі), що зберігає дані про користувача: user_id, ім’я, нік, поле для ролі: моодератор/адмін та додаткова інформація.

states - важлива сутність, що зберігає стан юзера за user_id. В таблиці зберігаються дані про номер тесту, запитання. На дані в цій таблиці додаток опирається, щоб запустити подальшу логіку для користувача.

Groups/ users_groups – сутності, що зберігають в собі дані про групи і дані про зв’язок користувачів з групами та дані про ролі користувача в цій групі (студент або куратор).

Tests/ questions – сутності, що мають в собі дані про тести: їх назву, який користувач створив, а для запитання: до якого тесту прив’язане питання, тип запитання, яка правильна відповідь, час, що відведено на запитання.

answers - сутність у якій кожен запис містить у собі відповідь на певне запитання, що ідентифікується за question_id. Також присутні дані продату відправлення запитання, дату завершення (або ні) ,за який час виконав завдання студен і чи вірно він його виконав, також id поточного тесту, дані про який можна переглянути у таблиці done_test. У цій сутності хберігається інформація про будь-які тести, зо колись були розпочаті. У сутності присутні такі поля: id поточного тесту, оцінка, дата старту та завершення тестування для користувача, пройдений тест чи ні (виставляється по завершенню тесту)



Рис. 2.2 ER діаграма

2.2. Вибір середовища розробки

IntelliJ IDEA Ultimate Edition компанії JetBrains [17] було обрано як середовище для розробки додатку. З огляду на численні рекомедації програмістів, засоби налогодження(Debug – режим), оптимальний графічний інтерфейс, зручна інтеграція з Maven, що автоматично підкачує усі залежності та бібліотеки, легке підключення до бази даних напряму з середовища,

інтеграція з системою конролю версій, що дозволяє виконувати комміти напряму з середовища. Наявність позитивного досвіду роботи з WebStorm та Commuity версією IntelliJ IDEA також від JetBrains вплинуло на мій вибір, так як продукти

мають схожий інтерфейс, що дозволяє мені швидше адаптуватися до роботи в цьому середовищі. На рис. 2.3 можна побачити приклад графічного інтерфейсу середовища з брейкпоінтами для дебаг режиму.

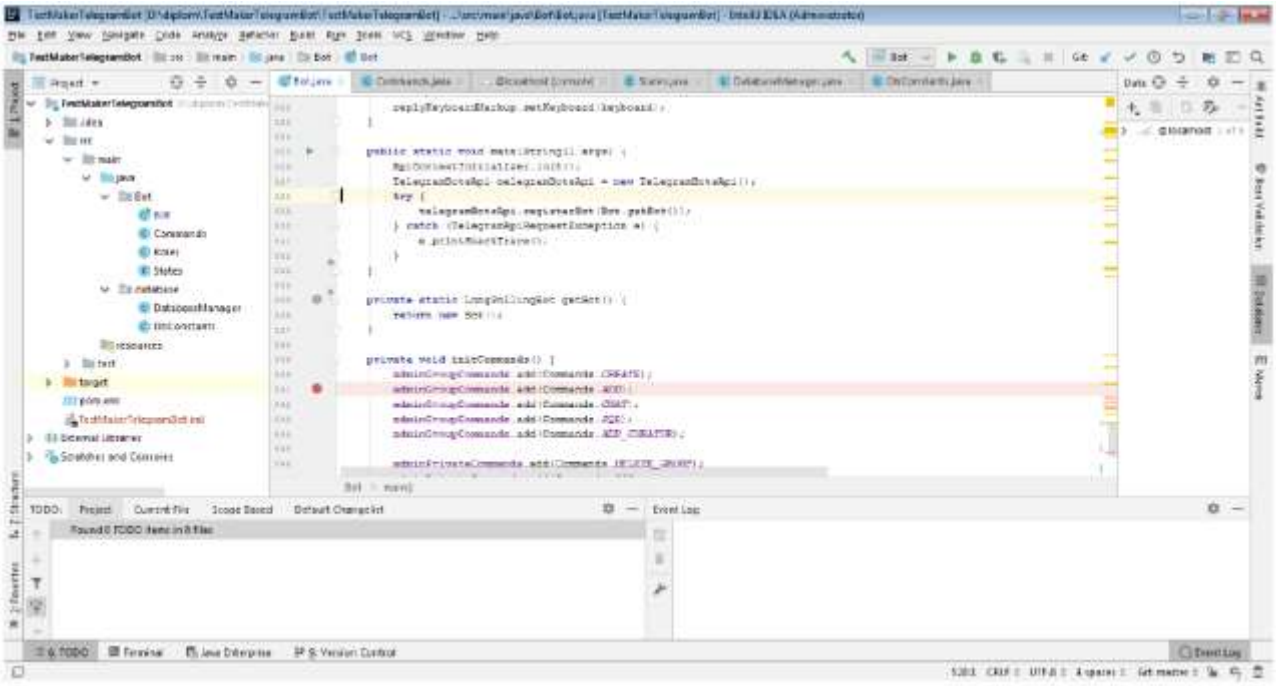


Рис. 2.3 Зовнішній вигляд середовища розробки

2.3. Визначення команд управління системою

Для даного додатку [20] можна визначити 4 основні ролі, які може мати користувач і один користувач може мати декілька ролей. Цими ролями є Звичайний користувач, Адміністратор, Студент групи та Куратор групи. Можливості цих ролей наявно показано на діаграмі можливостей на рисунку 2.4

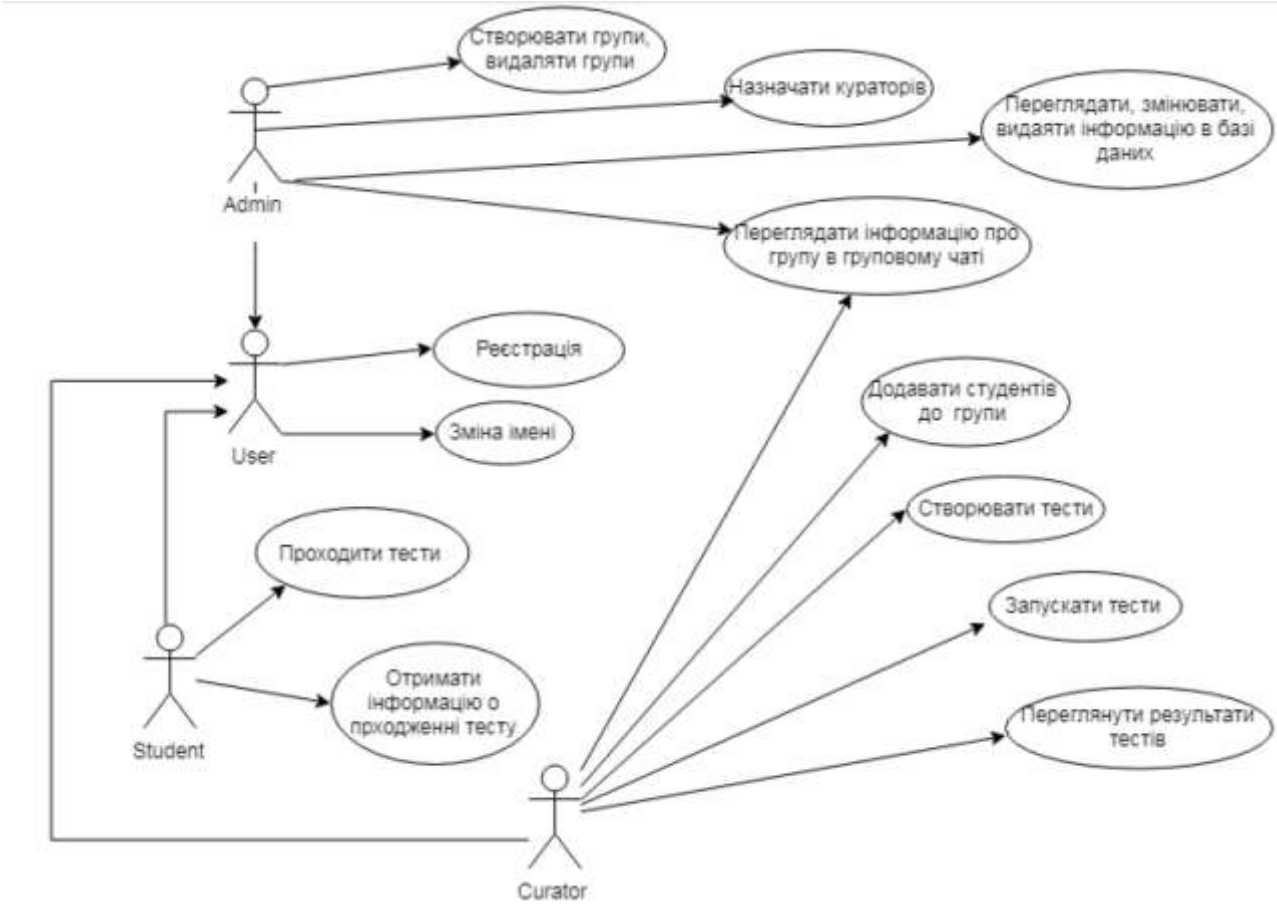


Рис. 2.4 Діаграма можливостей

Команди для телеграм-боту зазвичай починаються зі знака слешу “/” англійською мовою. Визначимо список команд для кожної дії користувача, які вказані на рисунку 2.4.

/change_name – команда для зміни імені, введеного при реєстрації, при введені імені довшого за 200 символів, система його обріже до потрібного розміру

/sql – команда лише для Адміністратора, наприклад /sql select * from users – покаже усі дані про користувачів, зареєстрованих у системі

/create – для створення групи, наприклад /create group2a

/delete_group – для видалення групи для даного групового чату

/chat – для перегляданння інформації про групу

/add_curator – назначає куратора групи, використовується відповіддю(reply) на повідомлення потрібного користувача

/add – додає користувача студентом до групи, використовується аналогічно команді /add_curator

/create_test – доступна лише користувачам, які являються кураторами хоч однієї групи, при натисканні на неї створюється тест і надається можливість додавати до нього запитання

/add_question– для додавання запитання у тест, id – test_id потріббного тесту, команда генерується самостійно системою і користувачу непотрібно прописувати її самостійно

/show_tests – команда, яка виводить усі тести, створені куратором і генерує команду /start_test для кожного тесту

/start_test_[id] – для початку тесту усіх студентів заданої групи, id – test_id потріббного тесту, аналогічно генерується ботом командою /show_tests

/show_results_[id] – де id test_id потріббного тесту,генерується командою /start_test і виводить результати усіх студентів даної групи, які проходять/пройшли даний тест

2.4. Визначення основних методів програмування додатку

Коротко основи роботи телеграм-ботів [20] можна пояснити принципом запит-відповідь, але так як нам потрібно сформувати більш складну систему зі зберіганням даних, на рисунку 2.5 можна переглянути основну схему роботи системи.



Рис. 2.5 Принцип роботи додатку

Основною особливістю від звичайних десктоп або веб-додатків являється неможливість крок-за-кроком логіки, телеграм Апі не зберігає сесію

користувача, не можна запрограмувати поступово запитання-відповідь послідовність. Системі просто приходить Update об'єкт який зберігає у загальну собі інформацію про користувача та повідомлення, яке він надіслав. Тому у пункті 2.4 був створений список команд, які дозволяють системі дізнатися, яку інформацію потребує користувач, але такі дії, як ввести своє ім’я, надіслати відповідь на запитання важко оформити в мову команд і це буде незручно для користувачів, тому для даного додатку реалізований принцип збереження станів. Для збереження стану буде використовуватися база даних і по занесеному стану у базі система зрозуміє, як саме обробляти подані до неї дані. Також, так як специфіка додатку полягає у проходженні тестів, для користувача, який проходить тест буде зберігатися унікальній ідентифікатор тесту (id) та запитання.

2.5. Адаптивний метод підбору часу

Тестування студентів в режимі онлайн потребує певних обмежень у часі, які задає викладач. Проте ми стикаємося з проблемою, що людина не може коректно оцінити час, який потрібен студенту для виконання завдання: деякі можуть бути занадто складні, а інші навпаки вирішуватимуться миттєво.

Для даного додатка постала задача: вибрати і реалізувати певний алгоритм, який на основі того, як завдання вирішували інші студенти, підбере оптимальний час для виконання завдання для наступного студента.

В п.1.4 було описано декілька методів, які можна використати для вирішення задачі. Якщо ми візьмемо глибоке навчання, то вхідним шаром будуть дані про те як студенти виконали завдання, а на виході ми маємо отримати найбільш оптимальний час. Для масштабної програми, де тести проходять мільйони користувачів цей метод був би оптимальним, але в нашому випадку ми не матимемо стільки даних, яких система потребує для навчання. Також ми розглянули IEI-технологію.

Проаналізувавши сет вхідних даних та специфіку застосування(нам не потрібна велика точність), було додатково розглянуто декілька способів для реалізації поставленої задачі. Було вирішено написати власний більш простий алгоритм. Так як програмно є можливість задавати час, за який студент може виконати завдання, то ми маємо такий вхідний сет для одного завдання: стартовий час(час, який був наданий студенту на виконання завдання); час, за який студент виконав завдання; чи встиг дати відповідь взагалі; чи вірно відповів на запитання. На виході ми повинні отримати як зміниться час, за який студент має відповісти на запитання див. рис. 2.6.

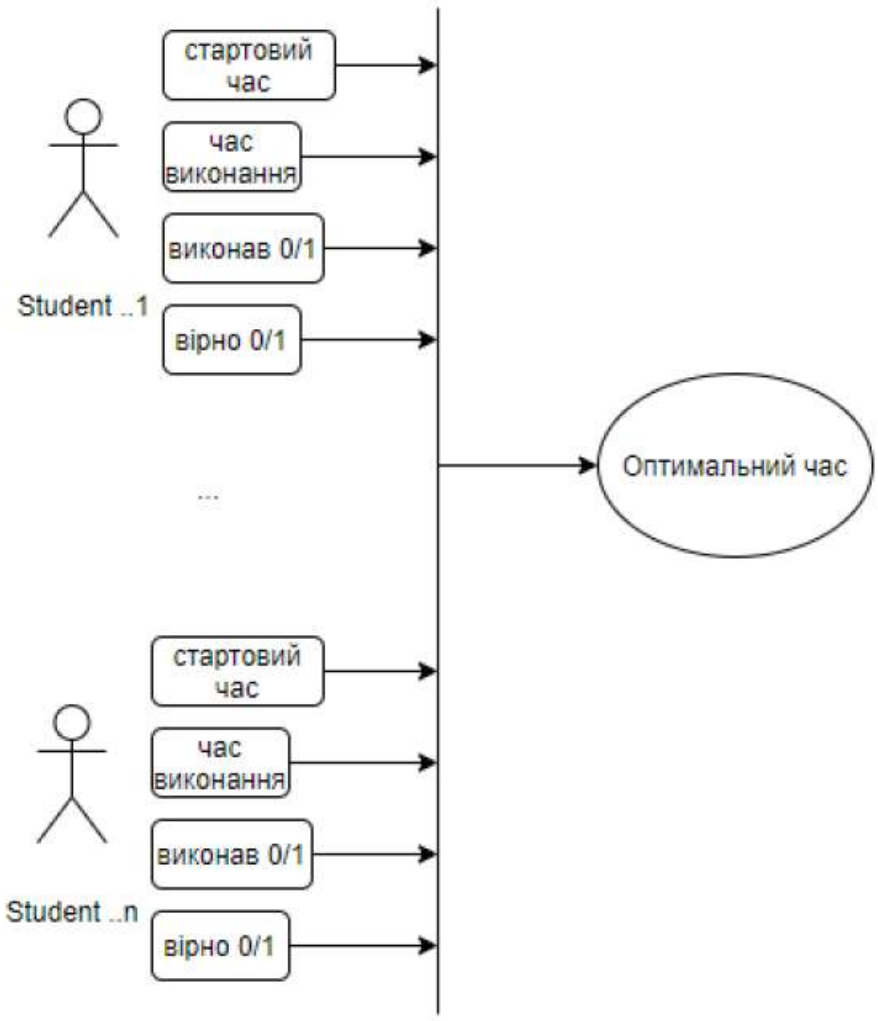


Рис. 2.6 Вхідні та вихідні дані

Спрощену схему алгоритму можна переглянути на рис. 2.7, де на вхід подаватиметься сет даних і в циклі буде проводитися обробка кожного студента

і в кінці, на основі первинного аналізу, буде підраховуватися коефіцієнт, який потім помножиться на стартовий час і ми отримаємо в результат час, що нам потрібний.

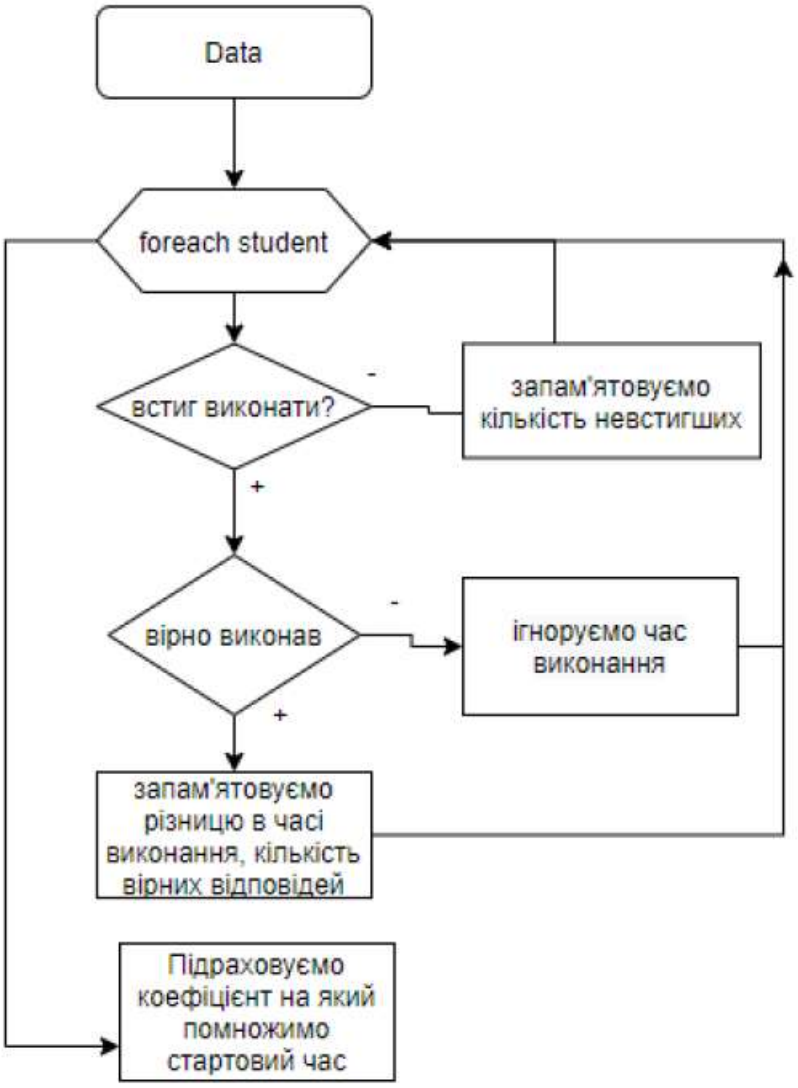


Рис. 2.7 Спрощена схема алгоритму

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1. Створення бази даних

Особливість MariaDB СУБД – синтаксис CREATE table IF NOT EXISTS, що буде використовуватися для створення таблиць для усунення помилок при повторному запуску скрипта.

Таблиці states та users можно переглянути в таблиці 3.1.

Таблиця 3.1

Опис атрибутів сутностей users, states

Таблиця	Поле	Тип данних	Ключі	Обмеження
Users	user_id	INTEGER	PK	Не пустий
	name	CHAR (100)		
	nick	CHAR (30)		
	role	CHAR (30)		
	description	TEXT		
States	user_id	INTEGER	PFK - users	Не пустий
	state	CHAR2(100)		Не пустий
	cur_test_id	INTEGER	FK – answers	
	question_id	INTEGER	FK - answers	

id - унікальний ідентифікатир запису у сутностях tests , groups, done_tests, questions, який генерується автоматично за допомогою СУБД, синтаксично описується за допомогою лексеми AUTO_INCREMENT[12]. Поля цих таблиць можна побачити у таблиці 3.2.

Таблиця 3.2

Опис атрибутів сутностей groups, tests, questions, done_tests

Таблиця	Поле	Тип данних	Ключі	Обмеження
groups	group_id	INTEGER	PK	Не пустий, auto_increment
	name	CHAR (100)	Унікальний	name
Tests	test_id	INTEGER	PFK - users	Не пустий
	name	CHAR(100)		Не пуст
	user_id	INTEGER	FK – answers	
questions	question_id	INTEGER	PK	Не пустий
	test_id	INTEGER	PFK - tests	Не пустий

Продовження таблиці 3.2

	question	VARCHAR(2000)		
	type	CHAR2(330)		
	data	TEXT		
	answer	VARCHAR(2000)		
	filepath	CHAR2(200)		
	timer	INTEGER		
done_tests	cur_test_id	INTEGER	PK	Не пустий
	test_id	INTEGER	FK - users	Не пустий
	user_id	INTEGER	FK - tests	Не пустий
	start_time	DATE		
	end_time	DATE		
	mark	INTEGER		
	passed	INTEGER		

В таблиці questions для реалізації таймерів було створено поле timer, де зберігажться час, який дажться студенту на запитання.

Для таблиць answers , users_groups, унікальний ідентифікатор не задано, первинний ключ в цих сутностях – складовий, див. деталі у табл. 3.3.

Таблиця 3.3

Опис атрибутів сутностей users_groups, answers

Таблиця	Поле	Тип данних	Ключі	Обмеження
users_groups	user_id	INTEGER	FK - users	Не пустий
	group_id	INTEGER	FK - groups	
	role	CHAR (100)	Не пустий	
answers	user_id	INTEGER	FK - users	Не пустий
	is_right	INTEGER		
	cur_test_id	INTEGER	PFK – done_tests	Не пустий
	question_id	INTEGER	PFK - questions	Не пустий
	timer	INTEGER		
	start_time	DATETIME		
	end_time	DATETIME		
	answer	VARCHAR(2000)		

В таблиці answers start_time – заповнюється при відправленні запитання користувачу, як sysdate(), end_time - sysdate() при відповіді на запитання

користувачем або, якщо користувач не встиг відповісти на запитання, то момент, коли час на відповідь сплинув.

3.2. Приклад типових запитів до бази даних

1. Вибір стану користувача за заданим user_id:

Select state from states where user_id = ?;

2. Вибір запитання і тесту, на якому в даний момент зупинився користувач:

Select cur_test_id, question_id where user_id = ?;

3. Вивести всіх студентів у групі:

Select user_id from users_groups where group_id = ? and role = ?;

Де параметр role - ідентифікатор ролі студента, наприклад “STUDENT”

4. Вибір усіх оцінок і інформації о сдачі/не сдачі усіх тестів користувача

Select mark, passed from done_tests where user_id = ?;

5. Вивести дані о усіх тестах даної групи

Select mark, passed from done_tests

where user_id in (

Select user_id from users_group

where group_id = ? and role = ?);

6. Вивести усі дані про запитання для певного тесту

select QUESTION_ID, TEST_ID, QUESTION_ID, TYPE,

DATA, ANSWER, FILEPATH, TIMER

from QUESTIONS

where TEST_ID = ?

7. Запит для оновлення таймерів для запитань

update QUESTIONS set timer = ? where question_id = ?

8. Запит для обновлення даних при відповіді на запитання студентом

update ANSWERS set answer = ?, is_right = ?, end_time = sysdate(), timer = TIMESTAMPDIFF(SECOND, start_time, sysdate())

" where user_id = ? and cur_test_id = ? and question_id = ?

3.3. Програмна реалізація додатку

Код додатку розділяється на 2 пакети: модель та логіка боту.

Розглянемо model, package model.entities зберігає в собі основні сутності, якими маніпулює програма: користувачі, стани, тести, запитання, відповіді, виконані тести, можна докладніше переглянути на рис. 3.1.

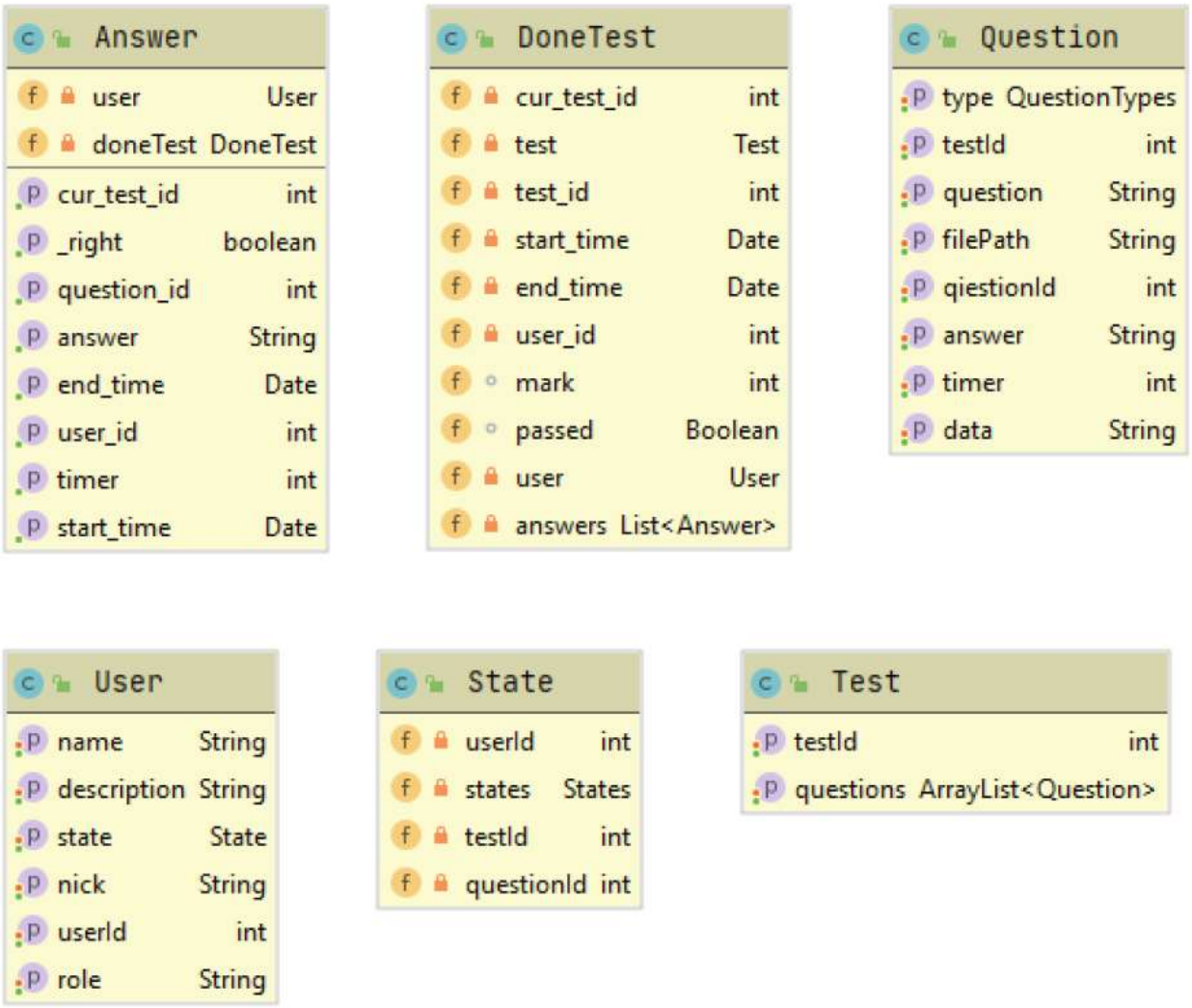


Рис. 3.1 Діаграма класів пакету model.entities

Розглянемо пакет model.database для роботи з базою даних, переглянути структуру пакету можливо на рисунку 3.2, змодельована даграма класів пакету.

Класс DoneTestAnswersDB включає в себе методи для маніпулювання сутностей: виконаний тест та відповіді, QuestionTestDB відповідно – завдання/тести, а UserStateDB для користувачів та їх станів.

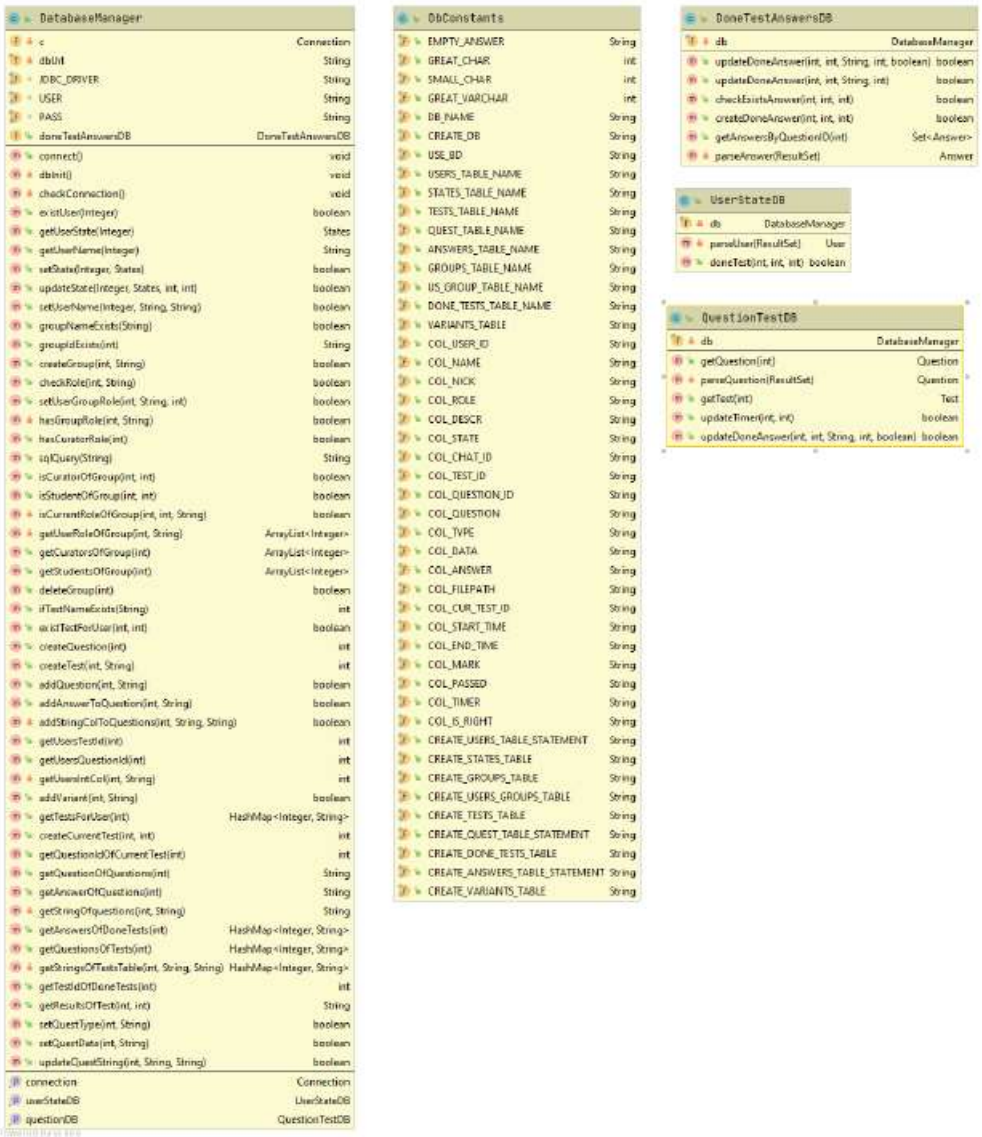


Рис. 3.2 Діаграма класів пакету model.database

Клас DbConstants зберігає у собі константи для моделювання запитів та роботою з базою даних

Константи GREAT_CHAR, GREAT_VARCHAR, SMALL_CHAR зберігають у собі обмеження на стовпці у таблицях відповідно 100, 2000, 30.

Константа DB_NAME –зберігає назву бази даних.

CREATE_DB – для створення бази даних з певним кодуванням.

USE_BD – для переходу на дану створену бд.

Константи, які закінчуються на TABLE_NAME несуть відповідне ім’я таблиці.

Константи, які починаються на COL_ - імена колонок у таблицях.

Константи, які починаються на CREATE_ - зберігають запит для створення відповідних таблиць, далі використовуються в DatabaseManager для ініціювання бази даних.

Клас DatabaseManager [20] містить у собі методи для роботи з базою даних.

Під час створення об’єкту класу в конструкторі ініціалізується змінна з рядком підключення до бд, в JDBC_DRIVER константі зберігається JDBC драйвер, в даному випадку це "org.mariadb.jdbc.Driver".

Основним методом являється void connect(), який генерує підключення дл бази даних, та визиває метод dbInit().

```
Class.forName(JDBC_DRIVER);
c = DriverManager.getConnection(dbUrl, USER, PASS);
dbInit();
```

Метод dbInit() ініціює базу даних, в ньому запускаються усі скрипти по створенню бази даних, переключення на цю базу даних та створення таблиць, якщо вони не існують.

У кожному методі класу, який маніпулює з базою даних, визивається метод checkConnection(), який перевіряє чи закрите підключення, та визиває метод connect(), якщо воно закрилось.

Для запитів був викорастаний PreparedStatement, який запобігає sql-ін’єкціям і дозволяє кешувати часті запити. Типовий запит:

```
String sql = "select " + COL_NAME + " from " + USERS_TABLE_NAME +
" where " + COL_USER_ID + " = ?" – де імена таблиці та стовпців взяті з класу DbConstants.
PreparedStatement preparedStatement = c.prepareStatement(sql);
        preparedStatement.setInt(1, userId);
```

Для результату використовується ResultSet.

```
        ResultSet resultSet = preparedStatement.executeQuery();
        if (resultSet.next()) {
        name = resultSet.getString(1);
```

```
}
resultSet.close();
preparedStatement.close();
```

Розглянемо пакет model.keyboards для роботи з апі телеграму для створенні спеціальних клавіатур (завдання з кнопками) структуру пакету можливо на рисунку 3.3 на даграмі класів.

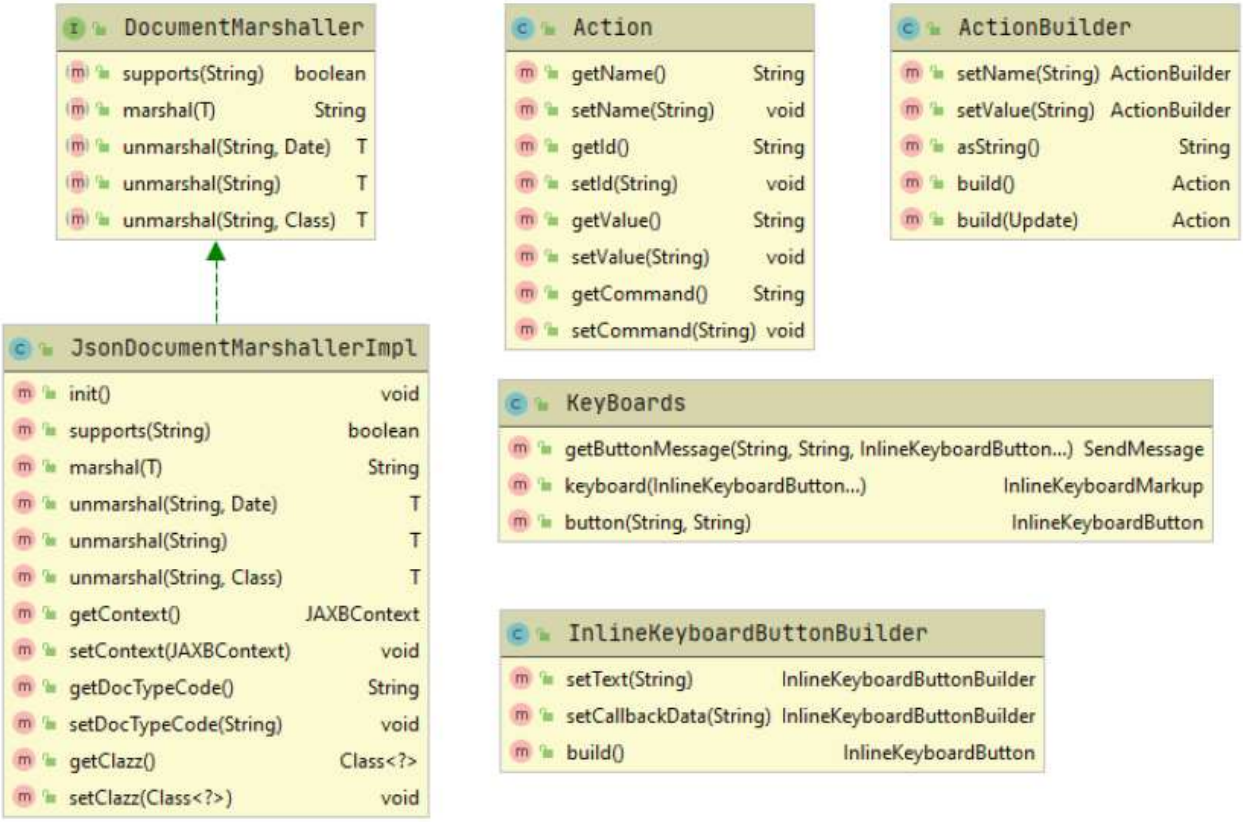


Рис. 3.3 Діаграма класів пакету model.keyboards

Розглянемо пакет Bot діаграму класів якого можна переглянути на рисунку 3.3.

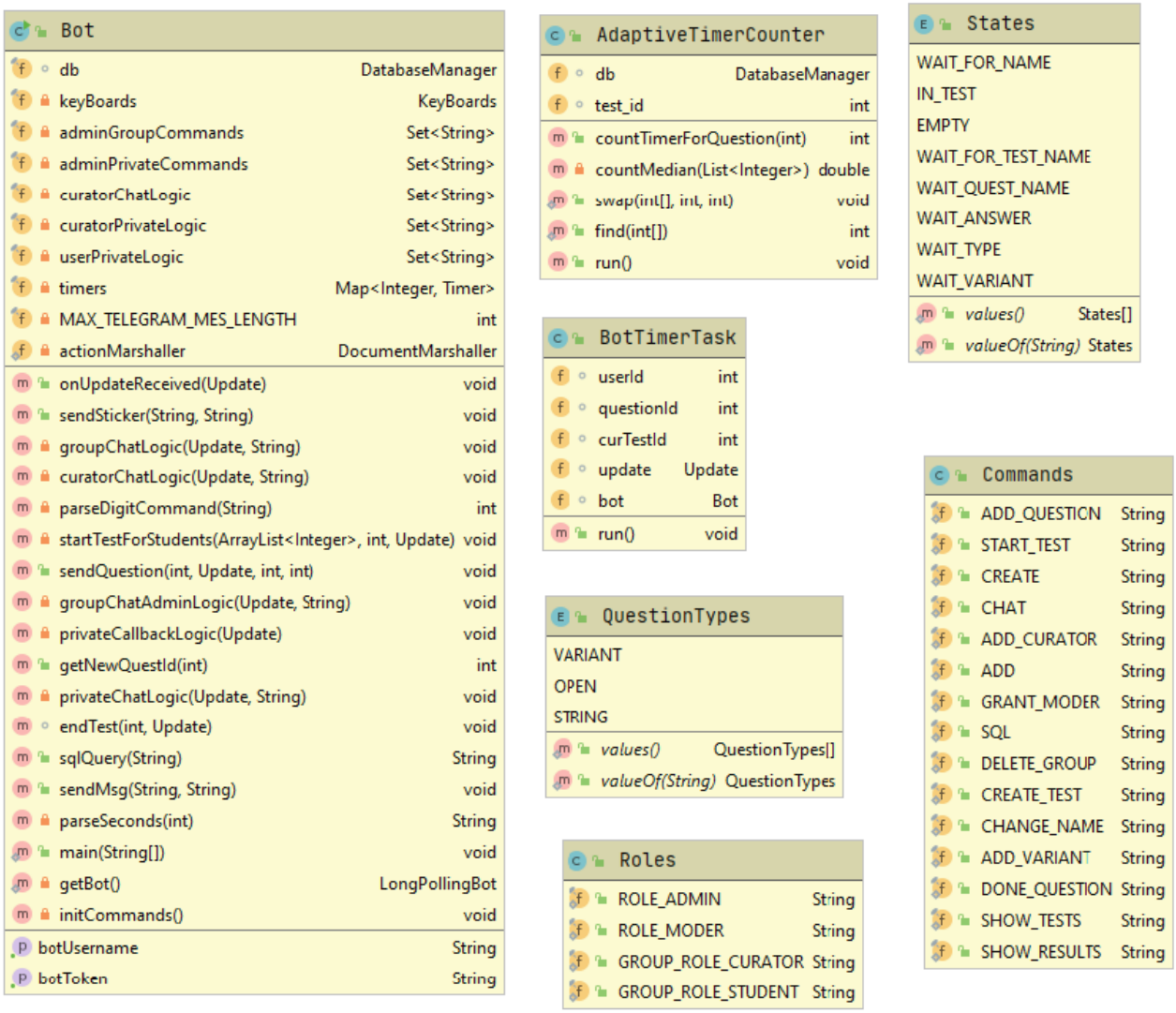


Рис. 3.4 Діаграма класів пакету Bot

Enum States створений для станів користувачів. Клас Commands зберігає константи з командами для бота, список команд ідентичний командам вказаним у пункті 2.4. Клас Roles містить у собі константи з основними ролями: роль адміністратора для користувача та ролі куратора і студента для групи.

Клас Bot наслідується від TelegramLongPollingBot, для цього потірбно перевизначити методи : getBotToken() - який повертає токен бота, getBotUsername() – що сберігає юзернейм бота. Дані в цих двух методах задаються/видаються BotFather в Телеграм. Також потрібно перевизначити метод onUpdateReceived(Update update) – який і реалізує основну логіку, він отримує обьект класу Update, який зберігає всю основну інформацію і вже потім виконує над ним якісь дії. Бот запускаться з функції main(), яка знаходиться в ньому же:

```
ApiContextInitializer.init();
        TelegramBotsApi telegramBotsApi = new TelegramBotsApi();
        try {
            telegramBotsApi.registerBot(Bot.getBot());
        }
```

У функції ініціалізується Апі контекст, створюється обьект класу TelegramBotsApi і запускається бот. Метод getBot() вертає новий обьект класу.

```
        private static LongPollingBot getBot() {
        return new Bot();
        }
```

В конструкторі бота створюється обьект бази даних, виконується підключення та ініціалізуються списки команд: adminGroupCommands, adminPrivateCommands, curatorChatLogic, curatorPrivateLogic в яких зберігаються команди для певних груп користувачів відповідно до назви Set.

```
Bot () {
        db = new DatabaseManager("jdbc:mariadb://localhost:3306/test");
        this.db.connect();
        initCommands();
        }
```

Після обробки даних і виконання команди користувачу має надійти відповідь, яка надсилається за допомогою методів sendMsg(String chatId, String s, Update update) в котрому рядок з повідомленням обертається в обьект класу

SendMessage, та обрізається на декілька повідомлень, якщо довше ніж 4096 сиволів – стандартне обмеження Телеграм на довжину повідомлення. Фінальне повідомлення відсилається за допомогою sendMessage(sendMessage) класу від якого наслідуємося.

Для реалізації таймерів у классі Bot було створено поле

```
private final Map<Integer, Timer> timers = new HashMap<>();
```

в якому зберігається id користувача та таймер, який запускався для нього при надсиланні запитання в методі sendQuestion:

```
if (timers.containsKey(userId)) {
timers.remove(userId);
}
Timer timer = new Timer();
BotTimerTask botTimerTask = new BotTimerTask(userId, questionId,
curTestId, update, this);
timer.schedule(botTimerTask, question.getTimer() * 1000);
 timers.put(userId, timer);
```

Де BotTimerTask це класс який наслідується від TimerTask і в run методі завершає поточне запитання та відправляє нове запитання або завершає тест, якщо поточне запитання було фінальним. Якщо ж система отримує відповідь на запитання до того, как сплине наданий час, то таймер відміняється:

```
 if (timers.containsKey(userId)) {
 timers.get(userId).cancel();
 }
```

Для реалізації адаптивного алгоритму розрахунку часових проміжків було створено клас AdaptiveTimerCounter, який наслідується від Thread.

У класі бот, при запуску тесту куратором стартується лічильник: AdaptiveTimerCounter counter = new AdaptiveTimerCounter(db,test_id); counter.start(); // start counter

В якому реалізована наступна логіка:

```
public void run() {
    Set<Integer> questions = db.getQuestionsOfTests(test_id).keySet();
    for (int quest_id:questions) {
        int timer = countTimerForQuestion(quest_id);
        db.getQuestionDB().updateTimer(quest_id, timer);
    }
}
```

Основна логіка підрахунку часу реалізована в методі:

```
public int countTimerForQuestion(int questionId) {
    Question question = db.getQuestionDB().getQuestion(questionId);
    int non_responded = 0;
    ArrayList<Integer> timediffs = new ArrayList<>();
    Set<Answer> answers =
db.doneTestAnswersDB.getAnswersByQuestionID(questionId);
```

```
for (Answer answer: answers) {
        if (DbConstants.EMPTY_ANSWER.equals(answer.getAnswer())) { if
(answer.getTimer() > question.getTimer()) { /* якщо людина не відповіла за
час, більший ніж поточний таймер, то підраховуємо його*/
non_responded++;
        }
} else {
         if (answer.is_right()) {
                timediffs.add(question.getTimer() - answer.getTimer());
         }
  }
}
double timediff = countMedian(timediffs); // підраховуємо медіану
int new_timer;
if ((double)non_responded/answers.size() > 0.3) { // якщо невідповівших більше
30%, то додаємо час
new_timer= (int)(question.getTimer() + timediff * ( (double)
question.getTimer() / timediff) * 0.5);
} else {
        new_timer = (int) ((double) question.getTimer() - timediff * (1 - (double)
non_responded / answers.size()) * 0.8);
        }
if (new_timer == 0) new_timer = question.getTimer();
return new_timer;}
```

Метод countMedian реалізований для підрахунку медіани, код можна додатково переглянути у додатку.

3.4. Приклад тестування програми та результату роботи з програми

На рисунку 3.1 можна переглянути процес реэстрації користувача

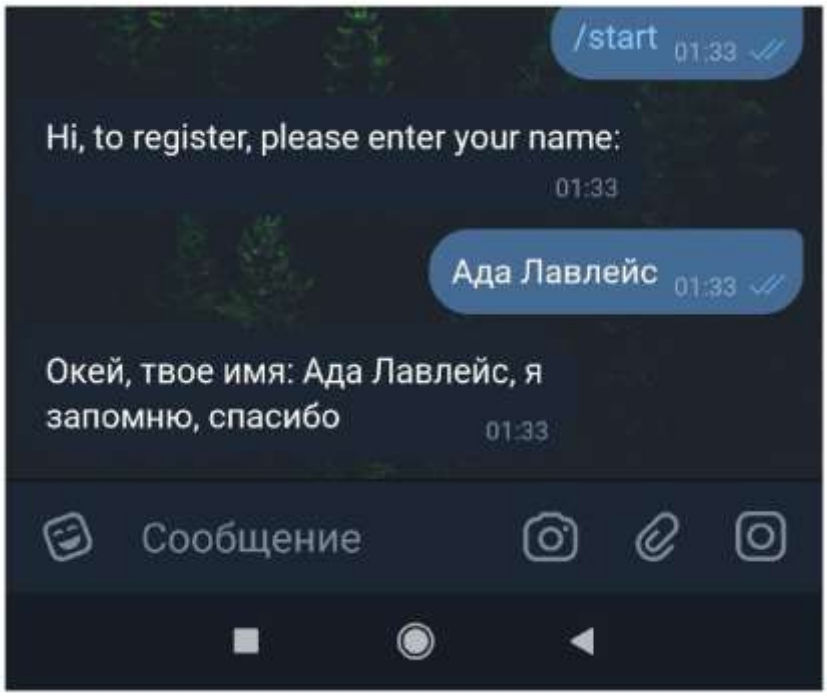


Рис. 3.1 Реєстрація нового користувача

Адміністрування чату можна побачити на рис 3.2 - 3.3, а саме створення чату, призначення юзерів кураторами, додавання незареєстрованих та зареєстрованих користувачів до групи.

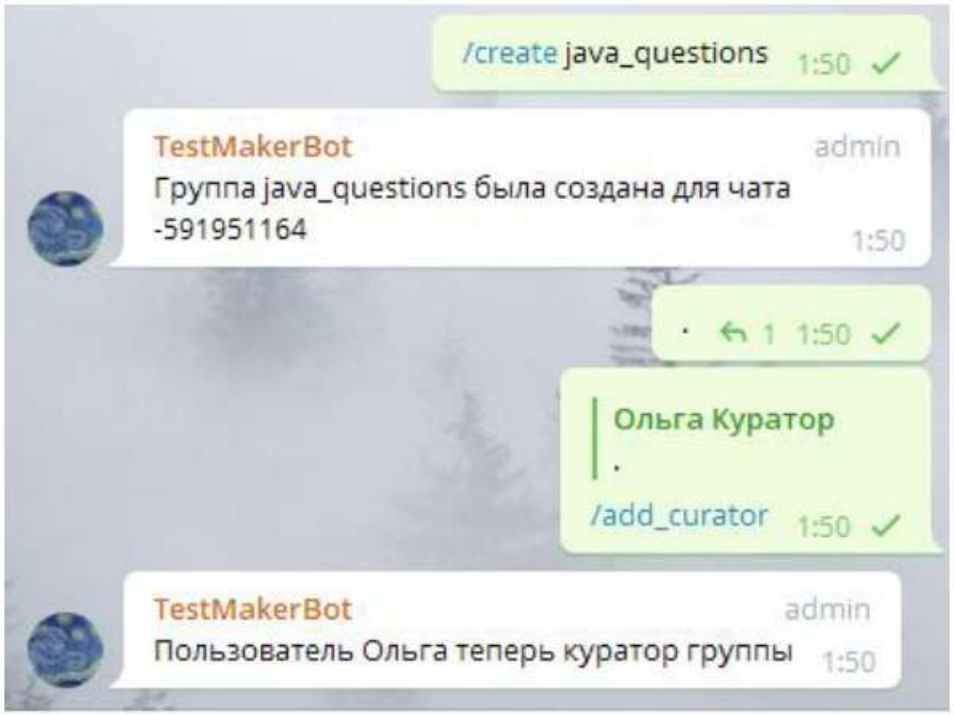


Рис. 3.2 Адміністрування чату, створення куратора

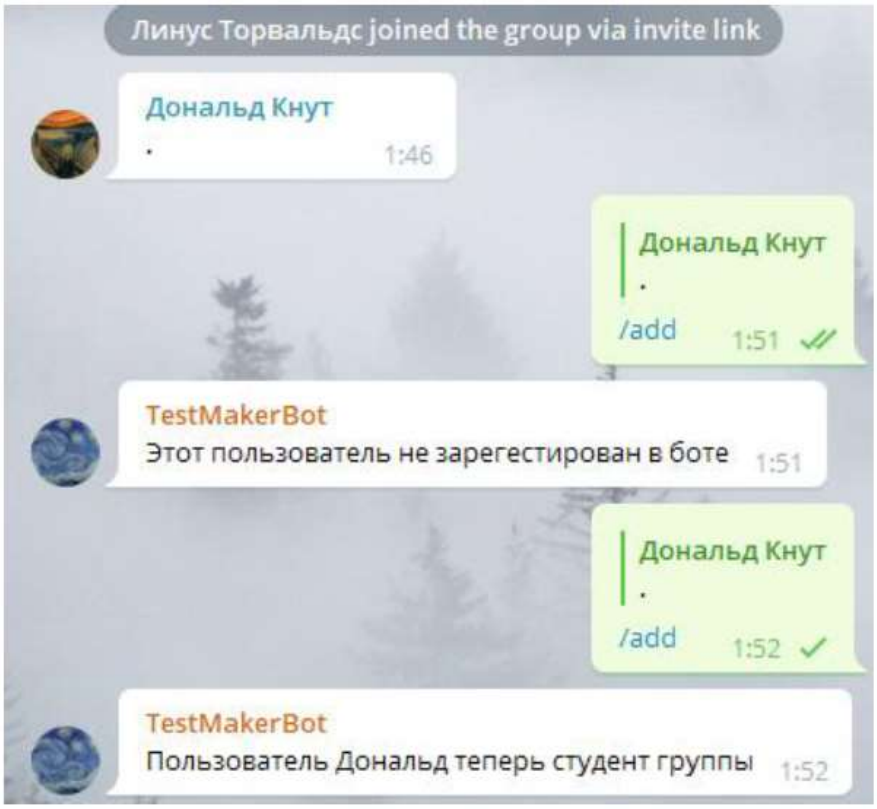


Рис. 3.3 Додавання нових студентів до групи



Рис. 3.4 Алгоритм створення тесту у куратора

Куратор будь-якої групи може створити тест з декількаами видами питань, на рисунках 3.4-3.5 ми бачимо створення опціональних запитань (відповіді кнопками) та строкових (відповідь являється словом або строкою).



Рис. 3.5 Додавання запитань(опціональні/строкові)

Тест починається куртором командою /start_test_ID, потрібну команду генерує бот, можна побачити на аутпуті команди /show_tests. Після закінчення тесту студентами, можна побачити результати командою /show_results_ID – рис 3.6-3.7.

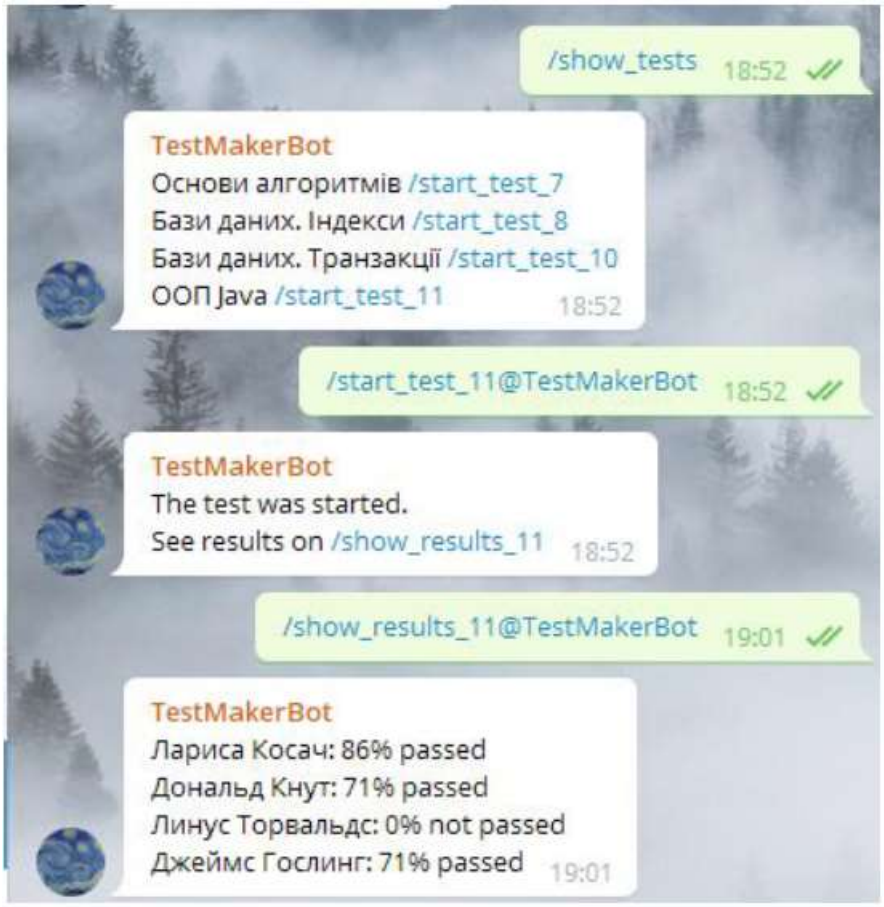


Рис. 3.6 Початок тесту і перевірка результатів

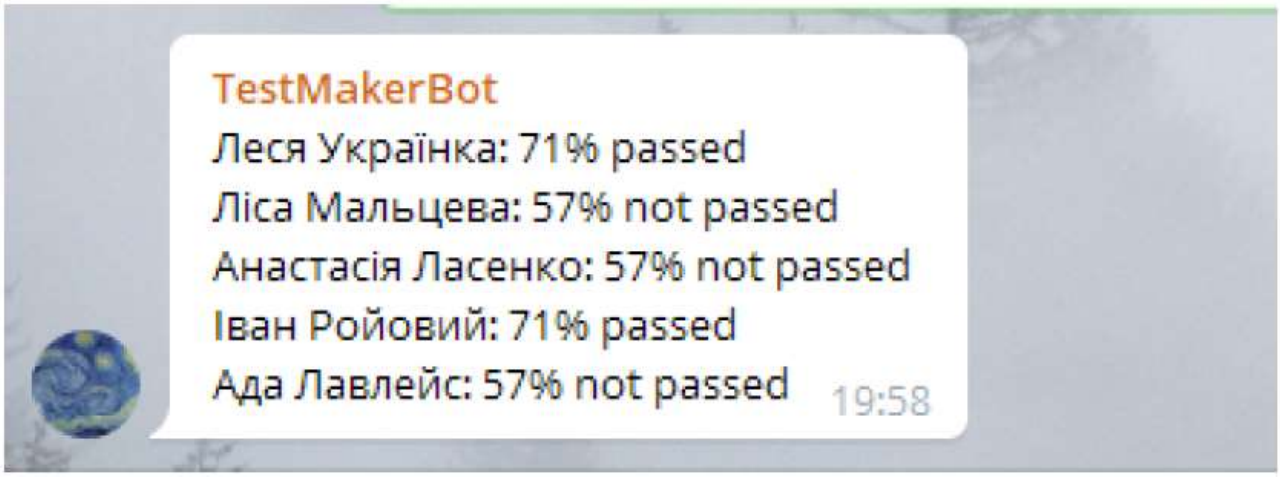


Рис. 3.7 Перегляд результатів для інших студентів

Якщо користувач не встиг відповісти на запитання (час, який відведений на запитання пишеться в тілі самого запитання, то бот надсилає відповідне повідомлення про те, що час сплинув (рис. 3.8) та надсилається наступне запитання. Запитання не зараховується студенту і враховується в результаті

тесту, як провалене. Після завершення тесту, користувач отримує повідомлення з балами за тест та провалив він його чи ні (рис. 3.9).

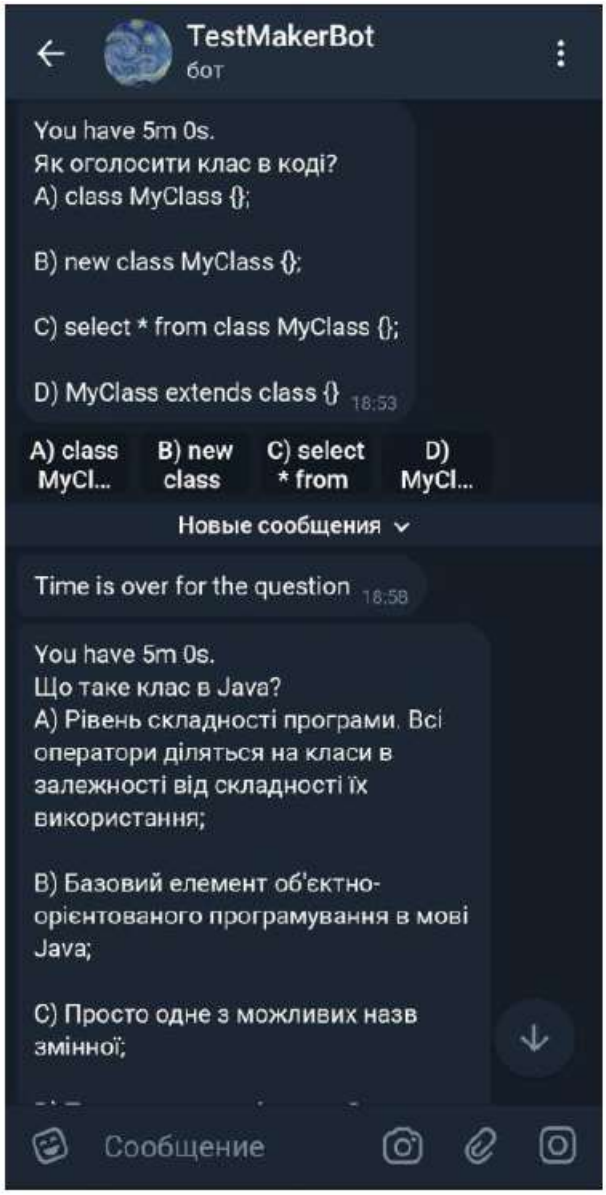


Рис. 3.8 Час сплив для користувача

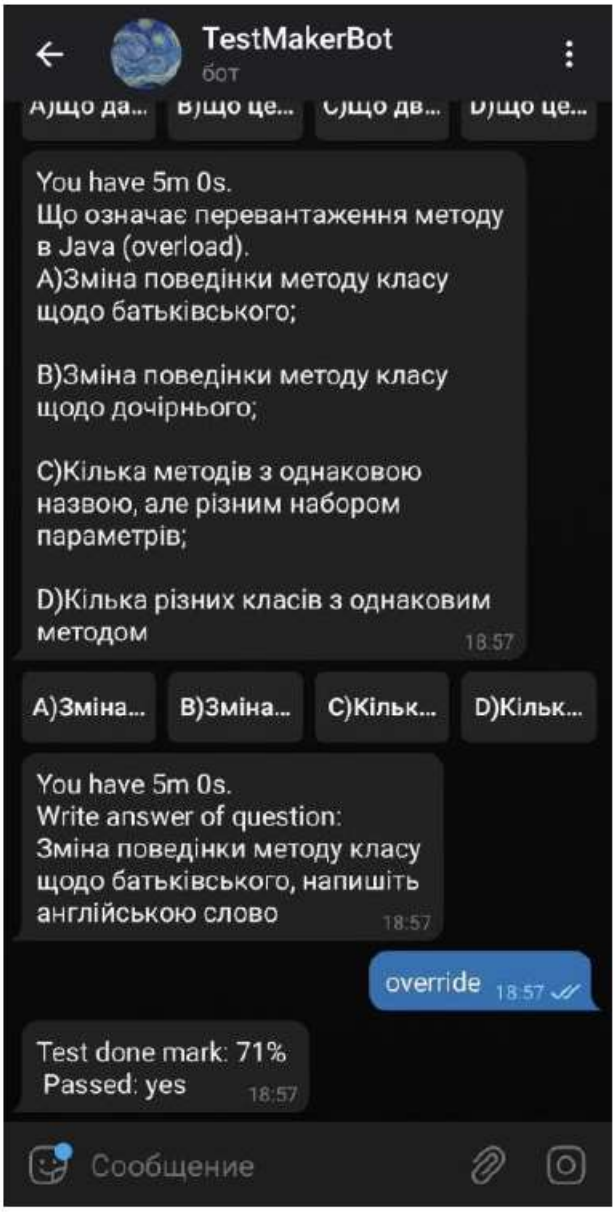
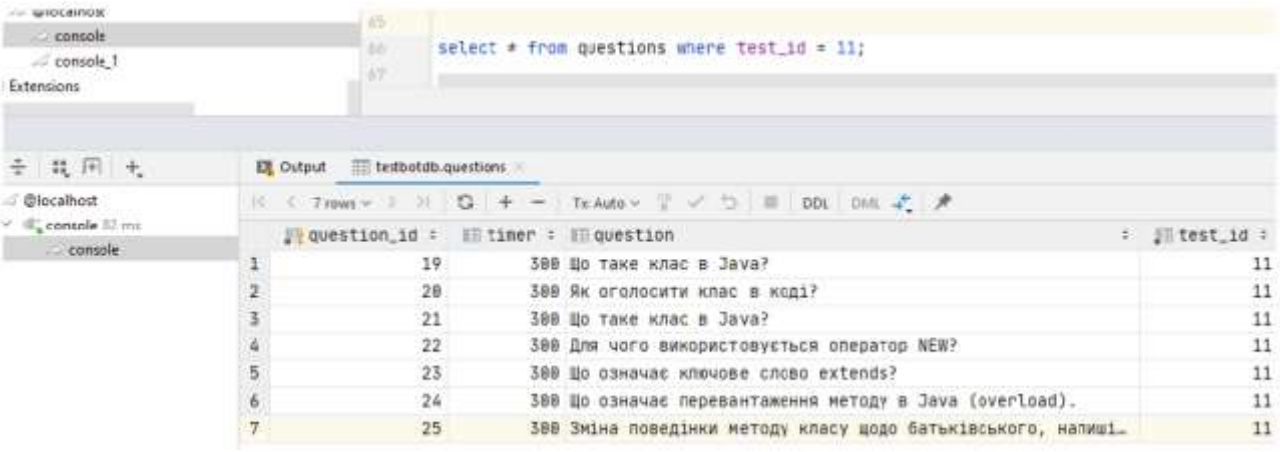


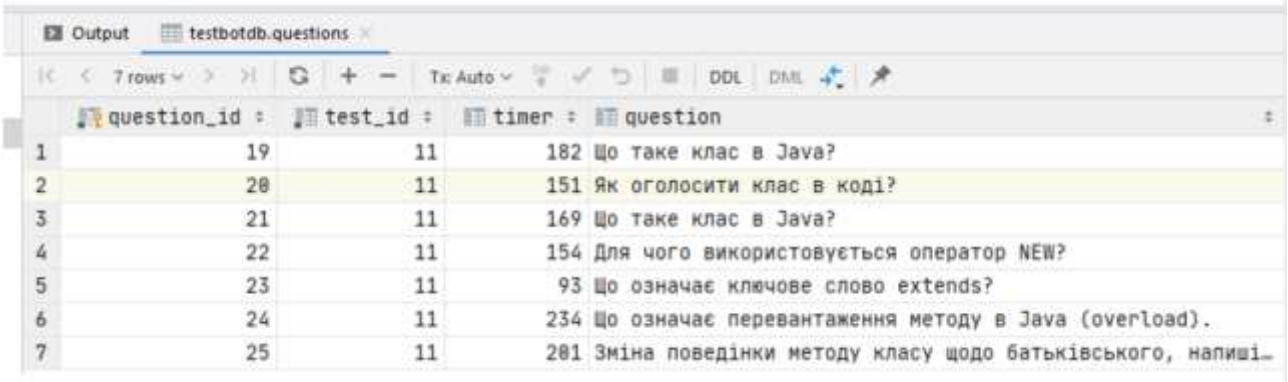
Рис. 3.9 Завершення тесту для користувача

Дефолтно задається час на запитання 5 хвилин(рис. 3.10), при кожному старті тесту перераховуються таймери для кожного запитання на основі того, як відповідали попередні студенти. На рис. 3.11 можна побачити як змінилися таймери через проходження тесту декількаами групами студентів.



	question_id	timer	question	test_id
1	19	300	Що таке клас в Java?	11
2	20	300	Як оголосити клас в коді?	11
3	21	300	Що таке клас в Java?	11
4	22	300	Для чого використовується оператор NEW?	11
5	23	300	Що означає ключове слово extends?	11
6	24	300	Що означає перевантаження методу в Java (overload).	11
7	25	300	Зміна поведінки методу класу щодо батьківського, напиші…	11

Рис. 3.10 Початковий список запитань для тесту



	question_id	test_id	timer	question
1	19	11	182	Що таке клас в Java?
2	20	11	151	Як оголосити клас в коді?
3	21	11	169	Що таке клас в Java?
4	22	11	154	Для чого використовується оператор NEW?
5	23	11	93	Що означає ключове слово extends?
6	24	11	234	Що означає перевантаження методу в Java (overload).
7	25	11	201	Зміна поведінки методу класу щодо батьківського, напиші…

Рис. 3.11 Зміна таймерів після проходжень тесту декількома групами

Також для користувачів з ролью Адміна доступна додаткова команда, яка дозволяє надсилати запити напряму до бази даних, приклад запиту можно побачити на рис. 3.12.

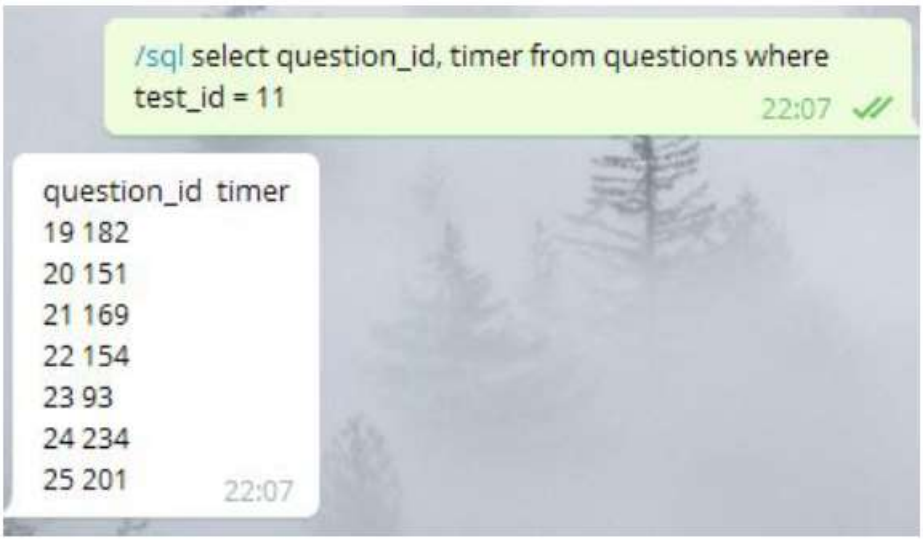


Рис. 3.12 Адмін-виконання команд в боті

ВИСНОВКИ

В випускній роботі був реалізований алгоритм для підрахунку часових проміжків, що виділені студенту на виконання завдання. Було проаналізовано декілька способів та можливостей для вирішення проблеми та обрано власну реалізацію алгоритму як найбільш зручну та підходящу на даному сеті вхідних даних та кількості цих вхідних даних.

Для зручності підрахунку, запуск перераховування таймерів починався при старті тесту в окремому потоці параллельно з початком тесту у студентів, що мінімізує можливість пересічення запитів та блокування даних у таблицях.

Обмеження часу було реалізовано за допомогою таймерів, що запускалися для кожного запитання і сет с таймерами зберігається в темповій пам’яті додатку, тоді як запитання, на яке має відповідати користувач, зберігається уже в базі даних.

Автоматизована система тестування на базі месенджеру Телеграм була реалізована на мові Java, було використано роботу з JDBC модулем (запити до БД), роботу Телеграм Bot Api і реалізовано всі вимоги до системи. СУБД для ІС – MySQL, до неї була реалізована ЕР діаграма з урахованням особливостей СУБД (наприклад автоінкрементні поля) та вимог до збереження даних про таймери.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Опис мессенджера Телеграм URL: https://tlgrm.ru/faq

2. Документація Телеграм по ботам URL: https://tlgrm.ru/docs/bots

3. Документація Телеграм по Bot API URL: https://tlgrm.ru/docs/bots/api

4. Вікіпедія, вільна енциклопедія, СУБД URL: https://en.wikipedia.org/wiki/Database#Database_management_system

5. Вікіпедія, моделі даних URL: https://en.wikipedia.org/wiki/Database_model

6. Інформаційний ресурс Habr URL: https://habr.com/ru/company/ruvds/blog/324936/

7. Інформаційний ресурс Володимира Драча URL: http://drach.pro/blog/hi-tech/item/145-db-comparison

8. Вікіпедія, Oracle Database URL: https://en.wikipedia.org/wiki/Oracle_Database

9. Інформаційний ресурс Llost URL: https://losst.ru/sravnenie-mysql-vs-mariadb

10. Knowledge Powerhouse, “Top 50 MySQL Interview Questions & Answers Kindle Edition”. Publisher “ Amazon Digital Services LLC”, 2016. 44p.

11. Інформаційний ресурс Tproger URL: https://tproger.ru/translations/sqlite-mysql-postgresql-comparison/

12. Hutten D., “MySQL: MySQL Tutorials for Beginners Basic to Advanced MySQL Languages”. Publisher “Amazon Digital Services LLC”, 2018. 122p.

13. MG Martin, "Java: Basic Fundamental Guide for Beginners Kindle Edition". Publisher “Amazon Digital Services LLC”, 2018. 79c.

14. 14. Colasoft Packet Player. URL: http://www.colasoft.com/packet_player/ (дата звернення 07.05 .2023)

15. Schildt H., “Java: A Beginner's Guide, Eighth Edition”. Publisher “McGraw-Hill Education”, 2018. 1152p.

16. Schildt H., “Java: The Complete Reference, Eleventh Edition”. Publisher “McGraw-Hill Education”, 2018. 1248p.

17. Yi R., Wang J., Tan P. Automatic fence segmentation in videos of dynamic scenes. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2016. P. 705–713.

18. Tariq Rashid, “Make Your Own Neural Network”. Publisher “Amazon Digital Services LLC”, 2016. 222p

19. Довбиш А.С., Основи проектування інтелектуальних систем. Сумський державний університет, 2009. 178с .

20. Автоматизована система тестування знань студентів на базі месенджера Телеграм, Бакалаврска робота, Литюга, О.Є. URL: https://essuir.sumdu.edu.ua/handle/123456789/74637