МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

Кваліфікаційна робота бакалавра

на тему: **Розробка десктопного додатку пошуку викладачів для додаткових занять**

Виконав студент групи **К-20і**
спеціальності 122 Комп'ютерні науки
Починок Павло Миколайович

Керівник **асистент**
Клепатська Вікторія Вікторівна

Консультант **д.т.н., проф.**
Казакова Надія Феліксівна

Рецензент **Начальник відділу впровадження інформаційних технології Департаменту інформації та цифрових рішень Одеської міської ради**
Корчемний Павло Анатолійович

Одеса 2022

ЗМІСТ

ТЕРМІНИ, СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

Веб-додаток – це програмне забезпечення або програма, яку можна відкрити за допомогою будь-якого браузера.

Десктопний додаток – це повнофункціональна програма яка розміщується на комп'ютері користувача.

Запис – це структура даних, що складається з фіксованої кількості взаємопов'язаних між собою компонентів одного або декількох типів.

Мобільний додаток – це програма призначена працювати на мобільних пристроях: мобільних телефонах, смартфонах, планшетах тощо.

Мова програмування – це спосіб передачі команд, наказів, чіткого керівництва до дії, які застосовуються для передачі компютеру інструкцій по виконанню обчислювального процесу і управління окремими пристроями та призначена для написання компютерних програм.

Налагодження – це процес пошуку і виправлення помилок в програмі, що перешкоджають коректній роботі.

Java – це мова програмування, розроблена компанією Sun Microsystems.

JVM – це програма, яка обробляє байтовий код і передає інструкції обладнанню як інтерпретатор.

Microsoft Visual Studio – лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. IDE MS Visual Studio .NET – це набір інструментів і засобів розробки різного роду застосувань (консольних, Windows, мобільних, Web-застосувань) та сервісів.

SQL – мова структурованих запитів декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми бази даних та її модифікації, системи контролю за доступом до бази даних.

ЗВО – заклад вищої освіти.

ІС – інформаційних систем.

ПК – персональний комп’ютер.

ОС – операційна система.

КР – кваліфікаційної роботи.

СКБД -Система керування базами даних.

IDE – Integrated Drive Electronics.

JDK – Java Development Kit.

MS – Microsoft.

MSDN – Microsoft Developer Network.

QA – Quality Assurance.

ВСТУП

Під час пандемії COVID-19, а також у зв язку з воєнними діями в Україні заняття багатьох шкіл оперативно переведені в режим онлайн, це привело до деяких технічних труднощів у вчителів та негативно позначилося на успішності учнів. Існуючий рівень навчання учнів у деяких школах залишає бажати кращого та дітям має сенс брати участь у додаткових заняттях. Для гарантованого вступу до закладу вищої освіти (ЗВО) учням часто потрібна допомога кваліфікованих викладачів.

Тому виникла потреба у додаткових заняттях учнів з необхідних предметів у дистанційному режимі, завдяки чому учні зможуть отримати знання. До переваг наданих репетитором послуг зазвичай відносяться: можливість навчання у себе вдома, проведення занять за найкращими методиками, контроль пройденого учнем матеріалу. Насамперед, йдеться про зручний час та організацію роботи.

Метою роботи є розробка та реалізація десктопного застосунку пошуку викладача для онлайн-навчання з різних предметів, дисциплін.

Згідно з Положенням про дистанційне навчання, затвердженим наказом Міністерства освіти і науки України, дистанційне навчання реалізовується шляхом:

- додаток дистанційної форми як окремої форми навчання;
- використання технологій дистанційного навчання для забезпечення навчання в різних формах.

Слід визнати, що нові системи віртуального навчання спочатку збивали з пантелику, але сьогодні, досвідчені викладачі подолати труднощі, пов'язані з роботою в режимі онлайн.

Велика кількість учнів не можуть продовжувати навчання у звичайному режимі. Так прийшла ідея створення додатку для пошуку викладачів, головна ціль якого – допомогти українським дітям освоїти знання по різним предметам.

Школяри–волонтери, які готові допомогти дітям із України можуть зареєструватися у додатку. Невимушене спілкування та різноманітні ігри разом з однолітком дозволять освоїти іноземну мову та інші предмети швидко та легко.

Буде доцільно створити простір, де можна знайти безкоштовного репетитора, простір , який об'єднуватиме дітей-волонтерів із дітьми з України.

Додаток дозволить записатися на заняття як до кваліфікованого спеціаліста (викладача з досвітом роботи) так і школяра-волонтера за допомогою декількох кліків. Заняття проходитиме в онлайн форматі і для нього потрібні: ноутбук/комп'ютер, інтернет та додатки для проведення заняття – zoom, discord, google meets.

Учні та їх батьки зацікавлені у швидкому та зручному доступі до інформації. Цінність інформації у світі дуже висока. Роль розпорядників інформації найчастіше виконують бази даних. Бази даних забезпечують надійне зберігання інформації, структурованому вигляді та своєчасний доступ до неї. Практично будь-яка сучасна організація потребує бази даних, що задовольняє ті чи інші потреби щодо зберігання, управління та адміністрування даних.

Дана бакаларська кваліфікаційна робота складається зі вступу, 3-ьох розділів, висновків, переліку посилань з 9 найменувань. Повний обсяг роботи становить 103 сторінки та містить 2 таблиці і 30 рисунки.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Порівняння існуючих додатків пошуку викладача

Розробка додатків сьогодні є однією з найпопулярніших послуг на інформаційному ринку. Кількість користувачів додатків за допомогою різних платформ невпинно зростає.

Користувач завантажує додатки як з метою розважитися, так і оптимізувати виконання робочих задач та навіть вирішити нагальні бізнес-питання. Ціна за втрату часу є надто високою, тому сучасний споживач інфоресурсів має бути оснащений найкращим програмним забезпеченням для щоденного користування.

На сьогоднішній день існує три основних типи додатків:

- десктопні додатоки **(**додатки для робочого столу);
- мобільні;
- веб-додатки.

Програми для робочого столу зазвичай більш повні, ніж веб і мобільні. З ними відкривається більше можливостей, тому що вони розраховані на більш просунуті параметри, а також роботу з повноцінними мишею, клавіатурою і великим монітором. Мобільні додатки вважаються полегшеними версіями комп'ютерних програм, тому що користувач обмежений невеликим дисплеєм, а працювати з інтерфейсом може тільки за допомогою своїх пальців або стилуса. А швидкість веб-додатків безпосередньо залежить від швидкості вашого інтернету [1].

Варто зауважити, що одночасно в декількох форматах можуть існувати додатки, але не всі тільки деяки із них. До них можна віднести програму Adobe Photoshop для робочого столу, існує і додаток Adobe Photoshop Sketch для мобільного пристрою, що також дозволяє малювати і редагувати. Ще є веб-версія Photoshop Express Editor, яка дає можливість редагувати зображення можна без скачування додатка (окремого) на персональний комп’ютер (ПК) або мобільний пристрій. Ще одним прикладом може бути програма Microsoft Word, яку

можна встановити як на робочий стіл так і на мобільний пристрій, ну і звичайно користуватися в веб-форматі

Десктопний додаток – це повнофункціональна програма ,що розміщується на комп'ютері. Він працює ізольовано від інших додатків і вимагає наявності клієнта (людини), яка працюватиме з програмою. Взаємодія між клієнтом та десктопним додатком відбувається за допомогою стандартного інфтерфейсу.

Цей тип додатків для своєї роботи не вимагає підключення до інтернету, але має більш високу швидкість, їх інсталяція залежить від використовуваної операційної системи (ОС), оскільки десктопний додаток вимагає встановлення на кожний комп'ютер користувача, який бажає працювати з даним додатком. Microsoft Word, Excel, медіа-програвачі, ігри – в загальному вигляді всі програми, які інсталюються у нас на ПК, можна назвати настільними додатками.

Веб-додаток – це програмне забезпечення (ПЗ) або програма, яка відкривається за допомогою браузера. Частіше всього за допомогою HTML, CSS, Javascript, які підтримуються всіми брацзерами в наш час – розробляється зовнішній інтерфейс. Для написання серверної частини / Back-end, найчастіше використовується будь-яка інша мова програмування або фреймворк, наприклад Python, PhP, Ruby або Java.

Мобільний додаток – це програма, що створюється для роботи на мобільних телефонах, смартфонах, планшетах тощо. Ці додатки можна встановити самостійно або ж завантажити з онлайн магазинів: таких як App Store, Google Play, Windows Phone Store та інших, безкоштовно або за окрему плату. Мобільні додатки, як мінімум переслідують 2 цілі – розважити власника гаджету та полегшити його побут чи краще організувати робочі моменти.

Є мобільні додатки схожі на десктопні додатки: годинник, калькулятор, радіо – до прикладу. А ті що взаємодіють з інтернетом – це вже мобільні веб додатки.

Порівняння видів додатків наведено в таблиці 1.

Таблиця 1 – Порівняння видів додатків

Параметр	Desktop додаток	Web додаток	Мобільний додаток
1	2	3	4
Доступ до Internet	Не потрібен	Необхідний, виняток деякі додатки можуть тимчасово працювати автономно	Не потрібен для нативних додатків (калькулятор) і потрібен для додатку таксі
Інсталяція / оновлення	Додаток повинен бути встановлений або оновлений	Інсталяція для всіх користувачів – однакова, з одноразовим налаштуванням – реєстрацією	Зазвичай додаток потрібно завантажити з спеціалізованого магазину і його потрібно налаштувати під себе
Інтерфейс взаємодії	Стандартні інтерфейси, стандартна взаємодія	Різноманітний інтерфейс взаємодії. Плюси – різноманітність реалізації, мінуси, кросбраузерна сумісність. Хоча сьогодні це питання є нівельоване впровадженням стандартів	Стандартні інтерфейси, стандартна взаємодія
Сумісність з пристроями	Залежність від платформи. Виняток – Кросплатформені додатки.	У більшості випадком – від платформи незалежне. Залежність від браузера.	Залежність від ОС Android, Mac і в деякій мірі від браузерів теж
Анімація, графіка	Швидкий відгук, швидкодія	Повільніший відгук, пов’язаний з передачею даних по мережі інтернет	Відносно повільніший відгук
Медіа	Проблеми з аудіо і відео – не має	Трапляються проблеми, залежно від реалізації через Flash, Socket тощо	Підвантаження залежить від апаратних можливостей гаджету
Користувацькі налаштування	Присутній тільки функціонал встановлений у користувача	Індивідуальні налаштування можливо підвантажити через Internet	Не всі мобільні додатки налаштовуються на зразок Desctop
Пошук по контенту	Відсутній	Примутній	Відсутній
Розшарювання	Є при початковому, додатковому налаштуванні	Більшість веб-додатків налаштовані на спільний доступ	Доступ може бути спільний або індивідуальний на вибір користувача

Продовження таблиці 1

1	2	3	4
Розробка	Під кожну платформу є свої інструменти, найчастіше необхідно писати окрему версію під різні платформи	Все виконується на сервері. Кроссплатформно, потрібен тільки браузер, а софт на сервері багатоплатформовий.	Мобільний додаток розробляється під конкретну ОС платформу Android або Apple iOS
Масштаби	Незначні, переважна більшість програм узагалі розрахована на використання одною людиною	Зберігання документів на Google Docs і інші сервіси.	Є додатки якими можна користуватися одноосібно, а є масові
Тестування	Тестування здійснюється фаховими QA engineer, групою QA рідше спільнотою тестувальників	Відкритість додатку дозволяє залучити більшу кількість QA.. В результаті більше покриття тестами і більш швидке виявлення вразливостей і некоректної роботи ПЗ	Тестовий процес схожий з тестуванням Веб додатків
Автоматизація тестування	Рідко, коли автоматизується	Вільно автоматизується	Автоматизує

Для роботи десктопного додатку необхідні тільки достатні технічні ресурси комп'ютера, сам додаток і набір бібліотек (файлів) функцій для роботи з додатком.

Даний тип програмного забезпечення вважається більш уразливим, оскільки користувач має доступ до системних файлів десктопної програми так як залежить від необачних дій користувача.

На даний момент, існує велика кількість інформаційних систем (ІС), які пропонують послуги репетиторів. Нижче було проаналізовано найпопулярніші із них, а саме: buki, vash-repetitor, bestrepetitor, preply, tvoyrepetitor та repetitor.

Ресурс buki.com.ua (рис. 1) – на сьогоднішній день це дуже популярний ресурс, який об'єднує 10 тисяч викладачів зі 120 різних напрямків знань, які надають послуги у 140 населених пунктах України.

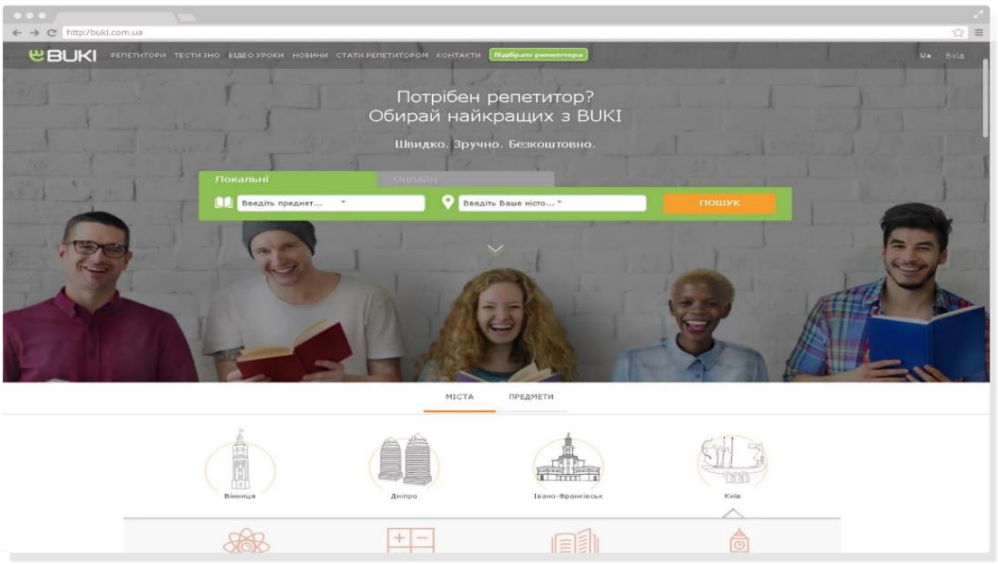

Рисунок 1 – Головна сторінка інформаційної системи buki

Анкети вчителів можна переглянути через пошукову панель. Слід обрати параметри фільтрування: предмет, рівень підготовки, місто, місце занять (на території репетитора, вдома в учня чи по скайпу). А також мінімальну та максимальну ціни, що для вас прийнятні.

Після того, як вибрали викладача, залиште заявку: він сам зв'яжеться з вами. Або ж вам зателефонує менеджер, щоб уточнити деталі. Підбір репетитора безкоштовний.

Наступна ІС, яка була розглянута – це зручний ресурс vash-repetitor.org (рис. 2)

Анкети репетиторів слід відфільтрувати за регіонами, предметами, вартістю послуг та рейтингом викладачів.

Рисунок 2 – Головна сторінка ІС vash-repetitor

Також є опція індивідуального підбору викладача за розширеними параметрами. Просто треба заповнити спеціальну форму. Потім протягом дня адміністратор повідомить телефоном результати пошуку та оптимальні варіанти. Консультації та підбір репетитора безкоштовні.

Спільнота об'єднує понад 220 тисяч викладачів у найбільших містах України та країн СНД. Адже займатися можна онлайн.

Один з успішних сайтів пошуку викладачів – preply.com (рис.3)

Доступні послуги викладачів з 55 предметів. Це шкільні та університетські дисципліни, а також хобі й захоплення. Є два шляхи пошуку репетиторів. Можна одразу ввести запит у рядок пошуку. А можна скористатися розширеними налаштуваннями. Для цього треба обрати напрям навчання, діапазон цін, бажаний графік занять, а також країну. Річ у тім, що тут доступний пошук викладачів – носіїв іноземних мов. Вони можуть жити за кордоном та надавати послуги онлайн.

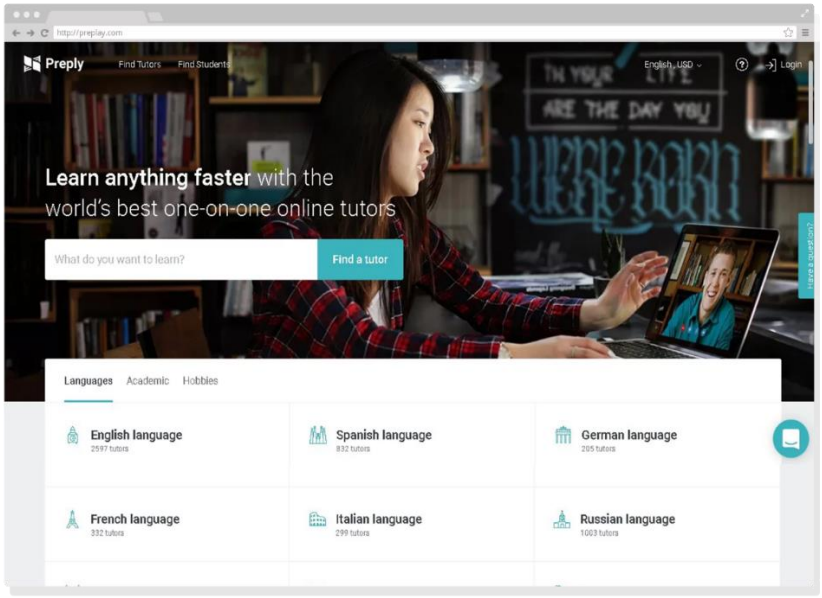

Рисунок 3 – Головна сторінка інформаційної системи preply

Тут можна одразу переглянути розклад репетитора та вільні дати для занять, щоб забронювати перший урок.

Анкети викладачів містять коротенькі відеопрезентації, що допомагають скласти перше враження про спеціаліста. Якщо виникають запитання, можна надіслати приватне повідомлення репетитору. Учні також можуть залишати оголошення на сайті. Вчителі переглядають заявки та відгукуються на ті, що їх зацікавили.

Один з відомих ресурсів repetitor.ua (рис.4). У нього немає панелі пошуку. Анкети викладачів упорядковані за двома основними параметрами: дисципліни та міста, у яких надають послуги репетитори. Однак у вільному доступі є не всі профайли, бо деякі вчителі їх закривають.

Більший вибір можна отримати, якщо скористатися розширеним пошуком. Тоді на вашу заявку можуть відгукнутися вчителі, які відповідають вашим критеріям, але не розміщували в публічному доступі свою анкету. Переглянути контактну інформацію викладача чи надіслати йому листа можуть лише ті, хто зареєструвався на порталі.

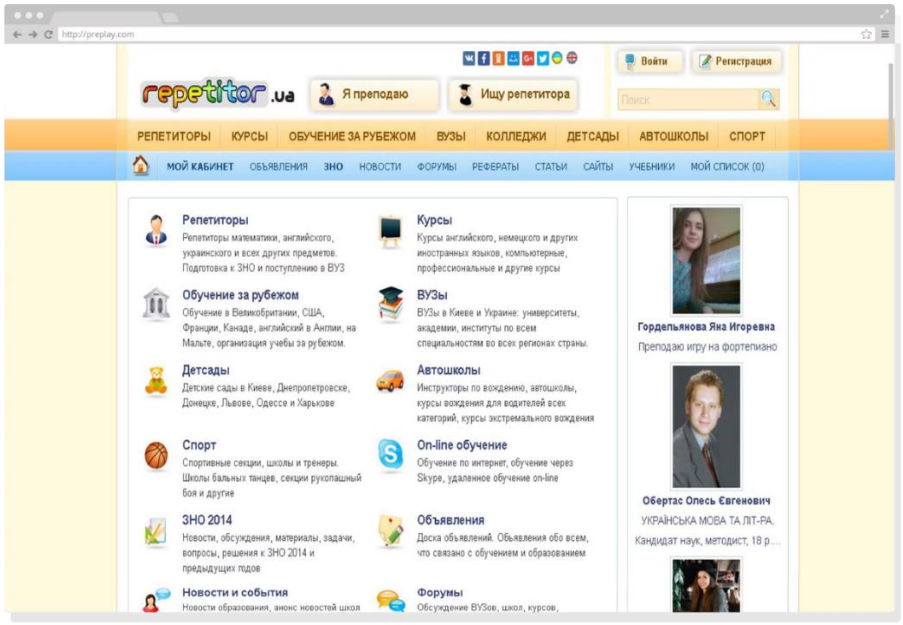

Рисунок 4 – Головна сторінка ІС repetitor

Ще один з багатьох сайтів, де можна знайти викладача – repetitor.org.ua

На сайті repetitor.org.ua реалізовано систему тестування викладачів. Панель пошуку містить два параметри: місто і предмет. За ними впорядковані всі анкети викладачів. Якщо певний вчитель відповідає вашим вимогам, необхідно заповнити доволі детальну анкету про потенційного учня та його побажання. Після цього надіслати заявку. На повідомлення відповідають безпосередньо репетитори. Однак, якщо часу на пошук немає, можна замовити підбір репетитора у менеджерів проекту. Ця послуга безкоштовна.

На сайті реалізовано систему тестування викладачів. Їм пропонують скласти тест із предмета, який вони викладають. Якщо репетитор пройшов перевірку успішно, у його профайлі з'являється відповідний бейдж. Це допомагає впевнитися в рівні підготовки репетитора. Також на сайті є великий розділ відгуків про вчителів.

Ще один сайт, який ми розглянемо це tvoyrepetitor. repetitor com.ua.

Тут є панель пошуку з кількома параметрами: предмет, місто, місце для занять чи уроки онлайн, стать репетитора і вартість його послуг. Коли позна-

чите важливі для вас пункти, отримаєте список анкет викладачів, які відповідають вашим вимогам. Можна також залишити заявку на підбір необхідного спеціаліста. Тоді менеджер обговорить з вами деталі телефоном, знайде репетитора і домовиться з ним про заняття. Лише після цього передасть вам контакти вчителя. У вільному доступі є статистика: скільки сьогодні опрацьовано замовлень від користувачів та скільки репетиторів було обрано.У базі близько 6 тисяч репетиторів з усіх регіонів України. Вони надають послуги з 82 предметів, а також хобі. Наприклад, можна знайти вчителя співу чи фотомистецтва.

Інформаційна система repetitor.com.ua має попит у користувачів. Тадиційної панелі пошуку чи бази викладачів тут немає. Необхідно зателефонувати за номерами, що вказані на сайті, чи приїхати безпосередньо на консультацію до студії. З кожним потенційним клієнтом спілкуються менеджер і методист. Вони визначають рівень знань і психологічний тип дитини, мету занять. Також запитують, чи зможе учень виконувати домашні завдання. Потім формують для нього графік і лише тоді пропонують викладачів на вибір. Учню обіцяють щомісяця надсилати звіт про успішність на електронну адресу.

1.2. Обгрунтування створення нового додатку

Даний додаток призначений для створення бази даних викладачів за різними науками.

Метою виконання проєкту є:

- набуття практичних навичок проектування баз даних та розробки програмного забезпечення з управління базами даних з використанням Visual Studio на мові програмування C#;
- створення власного програмного продукту засобами об’єктно-орієнтованого середовища мови програмування C#, та дослідження його складових компонентів.

Під час виконання КР було розроблено інформаційну базу даних викладачів, яка допоможе будь-якому користувачеві легко знайти потрібну інформацію з будь-якого предмета. Потужність бази даних обумовлена можливістю її постійного поповнення новими даними, причому у необмеженій кількості інформації. Це дуже зручно для користувача. Таким чином, створення бази даних, що має такі властивості, завдання досить актуальне і корисне.

Основні цілі:

- продавати та поширювати додаток;
- залучати клієнтів;
- поповнювати інформаційну базу даних викладачів;
- розповісти про види послуг;
- збирати Ліди (контактні дані);
- вибудувати довгострокові відносини з клієнтами;
- максимально адаптувати свій сайт під інтереси клієнтів.

1.3 Вимоги до функціональних характеристик додатку

Щодня 70% населення України в віці від 14 до 75 років виходять в мережу Google в пошуках викладача .Такі дані надає розроблений мною додаток.

Вимоги – швидко знайти потрібного викладача , який надасть можливості для якісного онлайн-навчаня. Система повинна працювати на сумісних персональних комп'ютерах, мати мінімальну конфігурацію.

Програмний продукт має володіти наступними функціональними характеристиками:

- мати можливість до подальшого вдосконалення;
- мати оптимальну функціональність;
- мати ергономічний інтерфейс;
- мати достатній рівень швидкодії.

Надійне та стійке функціонування програмного продукту має бути забезпечене шляхом:

- запобігання програмних збоїв та помилок;
- у разі виникнення програмного збою система повинна відновити роботу з останнього зафіксованого стабільного стану.

Розробка системи виконується в наступній послідовності, яка представлена в таблиці 2.

Таблиця 2 – Етапи робіт по створенню додатку

Етап	Зміст робіт	Результат
1	Розробка технічного проекту	Проектні рішення постворення додатку його частин. Узгодження проекту.
2	Реалізація технічного проекту (робочий проект)	Реалізація проектних рішень. Заповнення Пояснювальної Записки.
3	Тестування	Результати проведення тестів. Виправлення недоліків на стадії робочого проекту.
4	Здача-приймання КР	Оформлення документації. Передача проекту Замовнику.

2. РОЗРОБКА ТЕХНІЧНОГО ПРОЕКТУ

2.1 Обґрунтування вибору мов програмування

Мова програмування – це спосіб передачі команд, чіткого керівництва до дії, які застосовуються для передачі ПК інструкцій по виконанню обчислювального процесу і управління окремими пристроями та призначена для написання компютерних програм.

Мови програмування поділяються на 4 класи:

1. Низького рівня – це машинно-орієнтовані мови (асемблери, турбоасемблери, макроасемблери).

2. Універсальні мови високого рівня (BASIC, C, SCAL,TurboPASCAL).

3. Проблемно-орієнтовні мови – (динамо, GPSS, SPSS, побок) дозволяють відтворювати роботу певних виробничих підрозділів.

4. Мови зображення знань і роботи з ними (намети програм, які використовуються як тренажери при навчанні операторів).

За допомогою мов низького рівня створюють ефективні і компактні програми, оскільки розробник отримує доступ до всіх можливостей процесора.Мови низького рівня, як правило, використовують для написання невеликих системних програм, драйверів пристроїв, модулів стиків з нестандартним обладнанням, програмування спеціалізованих мікропроцесорів, коли найважливішими вимогами є компактність, швидкодія і можливість прямого доступу до апаратних ресурсів.

Далі проаналізовано найбільш популярні мови програмування для мікроконтролерів [2] Java, Асемблер, C, C++, C#, PHP, VisualBasic, Delphi та інші

Асемблер – це мова найнижчого рівня. Дана мова дозволяє найбільш повно розкрити усі можливості мікроконтролерів і отримати максимальну швидкодію і компактний код. У певних випадках не можна знайти альтернативу асемблеру, але при цьому асемблер має безліч недоліків. Незважаючи на

одержувану компактність машинного коду, програма, написана на мові Асемблер – громіздка і тчжка для розуміння. Щоб її створити необхідне досконале знання архітектури та системи команд мікроконтролерів.

Асемблер відмінно підходить для програмування мікроконтролерів, які мають обмежені ресурси, наприклад моделей з малим об'ємом пам'яті. Для великих програм краще використовувати інші мови, що відрізняються більш високим рівнем. Це дозволить створювати більш складні і при цьому зрозумілі програми.

Мови нижнього рівня,що орієнтовані на конкретний тип процесора і враховують його особливості, для перенесення асемблерної програми на іншу апаратну платформу необідно майже повне переписання. Певні відмінності є і в синтаксисі програм під різні компілятори.

C/C++ мови, були розроблені ще в 1983 році, дану мову використовували при створенні Microsoft Windows і Google Chrome. Завдяки широкому набору інструментів мова легко адаптується для додатків в різноманітних сферах життя, вне залежності чи то банківська сфера, розробка ігор, торгівля чи щось інше. Тому саме на цій мові можна створювати складні комерційні системи з багатьма елементами, а також розробляти прості додатки та програми. Мова C++ з широким інструментарієм та багатьма функціями. У C++ також легко зберігається процедурний стиль програмування.

Дано мова, C/C++, відноситься до мов більш високого рівня, в порівнянні з Асемблером. Програми на цій мові краще зрозумілі для людини. Перевагою C/C++ – є величезна кількість програмних засобів і бібліотек, що дозволяють просто створювати необхідний код. Фактично, C/C++ сьогодні стала основною мовою розробки керуючих програм. Компілятори даної мови реалізовані практично для всіх моделей мікроконтролерів.

C/C++ могутня і популярна мова програмування має і свої недоліки. Наприклад, важкі для розуміння і використання аспекти C++, пов’язані з керуванням пам’яттю і показниками. Мова C/C++ має досить складну для вивчення

структуру, а отриманний програмний код конкретного завдання, має більший обсяг, ніж код того ж завдання, реалізованого на Асемблері [3].

Pascal ще більш зручніша для сприйняття і вивчення мова. Проте, вона не має такого поширення як C/C++, особливо при програмуванні мікроконтролерів. Деякі окремі фірми підтримують цю мову, з метою спрощення переходу на контролери з великих ПК. Зокрема варіант мови під назвою MicroPASCAL входить до складу налагоджувальних засобів фірми Mikroelektronika.

BASIC – давня мова початкового рівня програмуванню, в даний час в основному збереглася у вигляді реалізації Visual BASIC від Microsoft. Використовується вона також для програмування мікроконтролерів. Реалізацій цієї мови набагато більше, ніж того ж Pascal. Пов'язано це в першу чергу з простотою мови. BASIC часто вибирають розробники програмно-апаратних платформ, націлених на спрощену розробку електронних пристроїв. Можна назвати такі проєкти, як PICAXE, Amicus18, microBASIC та інші. Недоліком BASIC є погана структурованість коду, її не варто вибирати для початкового вивчення з метою подальшого переходу на C/C++. Програмування мікроконтролерів на BASIC можна рекомендувати програмістам, які націлені на створення, в основному, простих пристроїв.

Javascript – використовується як одна з основних технологій для створення інтерактивних сайтів разом з HTML та CSS. Більшість браузерів використовують ці 3 основні технології. Також , використовуючи її , можна створювати ігри, мобільні додатки та десктопні програми.

SQL – мова структурованих запитів декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми БД та її модифікації, системи контролю за доступом до БД. Сама по собі SQL не є ані системою керування базами даних, ані окремим програмним продуктом. На відміну від дійсних мов програмування, SQL може формувати інтерактивні запити або ж, якщовона будована в прикладні програми, виступати як інструкції

для керування даними. Стандарт SQL містить функції для визначення зміни, перевірки та захисту даних [4].

SQL – це діалогова мова програмування для здійснення запитів та внесення змін до БД, а також керування БД. Багато БД підтримує SQL з розширеннями до стандартної мови. Ядро SQL формує командна мова, яка дозволяє здійснювати пошук, вставку, оновлення і вилучення даних за допомогою використання системи керування і адміністративних функцій. SQL також включає CLI (Call Level Interface) для доступу і керування базами даних дистанційно (див. рис. 6).

Основу бази даних SQL Server утворює сервер або ядро БД. Ядро БД відповідає за обробку запитів, які надходять від клієнтів, і передачу відповідних результатів клієнтським компонентам.

Рисунок 6 – Характеристики мови програмування SQL

Візуальні мови на відміну від класичних мов програмування, дозволяють розробляти програми у вигляді зображень. Серед таких мов можна виділити FlowCODE, Visuino, ArduBlock або Scratch. Перевагою візуальних мов є

зрозуміла структура алгоритму. Це дозволяє просто розібратися в його функціонуванні будь-якій людині, що знає основні символи мови. Переклад структурних схем в команди мікроконтролера, як правило, виконується не відразу. Спочатку алгоритм транслюється в команди асемблера або будь-якої мови високого рівня. Тільки потім, все – перетворюється в машинний код. Така схема, незважаючи на свою складність, дозволяє використовувати найбільш зручні компілятори різних розробників. Недоліком візуального підходу є громіздкість вихідних матеріалів. Проте, подібні мови програмування знайшли дуже велике поширення для вирішення спеціальних завдань.

Серед розглянутих мов програмування для розробки додатку у IntelliJ IDEA найбільш поширенішою є Java (рис.7).

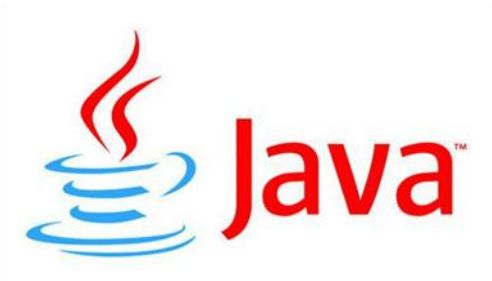

Рисунок 7 – Логотип мови програмування Java

Java є втіленням таких базових принципів:

- простота;
- безпека;
- перенесення;
- об’єктно-орієнтована направленість;
- стійкість щодо помилок;
- багатопоточність;
- незалежність від архітектури;
- інтерпретація;
- висока продуктивність;

- розподіленість;
- динамічність.

Створюючи застосунок за допомогою мови програмування Java, не слід замислюватися, в якій операційній системі працювати. Java має власний набір машинно-незалежних бібліотек – ще їх називають пакетами. Щодо процесорів ситуація така ж. Компілятор Java не генерує безпосередньо інструкції процесору, а створює проміжний код − байткод для віртуальної машини Java (Java Virtual Machine – JVM) Ядро віртуальної машини Java реалізовано практично для всіх типів ПК, вважаєть, що файли байткодів незалежні від платформи.

Java – це мова програмування, розроблена компанією Sun Microsystems. Мова програмування Java дуже популярна, оскільки 90% компаній використовують її в своїх розробках. Цю мову можна використовувати при розробці ОС Android , яка в даний час є найбільш мобільною платформою в світі. Цю мову можна використовувати для розробки десктопних додатків, ОС, back-end систем. Основна її перевага це кросплатформеність.

Програми на Java транслюються в байт-код, який потім виконується віртуальною машиною Java. JVM – це програма, яка обробляє байтовий код і передає інструкції, такому обладнанню як інтерпретатор. Перевагою такої реалізації є незалежність байт-коду від операційної системи і устаткування, що дозволяє виконувати Java-додатки на будь - якому пристрої, для якого існує JVM.

Іншою важливою особливістю технології Java є гнучка система безпеки завдяки тому, що виконання програми повністю контролюється віртуальною машиною. Операції, які перевищують встановлені повноваження програми (спроба несанкціонованого доступу до даних або з'єднання з іншим комп'ютером) викликають негайне переривання. В Java спеціально створили полегшену версію системного процесу– потік. Найбільшою проблемою, пов’язаною з процесами і потоками, є їхнє функціонування під керівництвом конкретної операційної системи.

Спеціалісти компанії Sun зробили потоки частиною мови програмування. Тому застосування, яке написане мовою Java, працюватиме в операційних середовищах Windows, Unix, MacOs.

Основний плюс – висока продуктивність Java . Інтерпретатор Java може виконувати байткоди зі швидкістю, яка наближається до швидкості виконання коду, відкомпільованого до машинного формату, що досягається завдяки використовування інтерпретатором багатьох потоків виконання. Java– це мова строгого використання типів, що зумовлює зменшення числа помилок під час написання програми. Тут відбувається автоматична перевірка виконання граничних умов. Також немає арифметики покажчиків і керування пам'яттю здійснюється автоматично. Програмний код, написаний мовою Java не може зіслатися на пам'ять поза простором програми або зробити помилку внаслідок вивільнення пам'яті і тим самим вичерпати всю пам'ять. Функції забезпечення безпеки дуже важливі для розподілених мереж з безліччю вірусів, троянських коней і т. п. Для реалізації цієї мети розробники мови Java створили механізм, який отримав назву пісочниці (sandbox):

- перевірку на рівні JVM;
- захист на рівні мови;
- інтерфейс Java Sucurity (цифрового підпису).

Інтерфейсом у мові Java називають абстрактний клас. Його введено для реалізації наслідування від декількох класів. HTML є засобом логічної організації інформації і створення гіпертекстових зв'язків з відповідними даними. Вона дає змогу читати документи не тільки зверху вниз, а й у будь-якому іншому порядку, проте ніколи не була мовою програмування. Єдиний зв'язок між HTML і Java − це наявний в HTML дескриптор APPLET, за допомогою якого викликається для виконання аплет Java .

Компанія JavaSoft, створена розробниками мови Java, пропонує безкоштовно набір засобів для програмістів мовою Java JDK (Java Development Kit) за адресою http:// java.sun.com/products/jdk/. JDK містить все необхідне для

створення програм: базові функції мови, інтерфейс прикладного програмування (API) з наборами пакетів й основні інструменти. Більшість версій JDK містять сім інструментів розробки на Java: 1) компілятор (javac); 2) генератор документації (javadoc); 3) генератор файлів заголовків і заглушок мови C++ для Java (javah); 4) інтерпретатор (java); 5) програму перегляду аплетів (appletviewer); 6) реасемблер файлів класів; 7) відлагоджувач програм [5].

Вище було проаналізовано найбільш популярні мови програмування для мікроконтролерів Java, C, C++, C#, PHP, VisualBasic, Delphi та інші.

Для розробки додатку у своїй КР скористався засобами об’єктно-орієнтованого середовища мови програмування C# .

Мова програмування C# (характеристики наведені на рисунку 8) вважається однією з найкращих. Цю об'єктноорієнтовану, універсальну й багатопарадигмальну мову досить легко вивчити. Знаючи цю мову програмування, можна працювати веброзробником, розробником ігор, розробником мобільних додатків або backend-розробником.

Рисунок 8 – Характеристики мови програмування C#

В той час, коли такі мови, як Python і PHP, існують досить тривалий час, C# вважається молодою мовою програмування. Данський інженер-програміст Андерс Хейлсберг розробив її у 2000 році.

Спочатку C# називалася COOL. Цей акронім в оригіналі походить від «C-style Object-Oriented Language», що означає «об'єктноорієнтована мова у стилі C». На жаль, компанія Microsoft не змогла зберегти цю «круту назву» (cool – англійською мовою має значення «крутий, класний» – Прим. ред.) через законодавство про торгові марки.

C# вимовляється «Сі-шарп». Назву взяли з музичної нотації, де символ «#» – октоторп або дієз – вказує на те, що ноту слід зіграти на півтону вище. Суфікс «шарп» також використовувався кількома іншими мовами програмування платформи .NET, а саме виданнями сучасних мов, наприклад, J#, A# та функціональна мова програмування F#. Базовий синтаксис C# подібний до мов стилю C, таких як C, C++ і Java. Ця мова програмування найбільше відповідає стандарту Common Language Infrastructure (CLI).

Так як створювати програму будемо для платформи Microsoft, то варто використати C#. Розробники віддають перевагу цій мові, оскільки вона добре продумана та проста у використанні.

C# використовується для:

- розробки сайтів. Ця мова дозволяє створювати динамічні вебсайти на платформі .NET або на програмному забезпеченні з відкритим кодом;
- розробки програм Windows. Оскільки C# створили у Microsoft, то вона частіше за інші мови використовується для розробки програм і додатків, специфічних для архітектури платформи Microsoft;
- розробка ігор. C# легко інтегрується з двигуном Unity. Завдяки мультиплатформності мову можна використовувати на будь-якому сучасному мобільному пристрої або консолі.

Список програм і додатків, написаних на C#, включає: Microsoft Visual Studio, Paint.NET, Windows Installer XML, Open Dental, FlashDevelop, KeePass, NMath, Pinta, Banshee, OpenRA.

Ось кілька причин працювати з цією мовою програмування:

- C# вважається простою мовою. Її компактний код легко читати, що надзвичайно зручно для оптимізації командної розробки програмного забезпечення;
- C# працює на платформі .NET, яка вважається надійною та добре спроєктованою;
- вона може заощадити час, оскільки ця мова була розроблена, щоб полегшити створення на її основі потужних інструментів;
- мова програмування C# масштабована і проста в обслуговуванні. Це мова з відкритим вихідним кодом, створена Microsoft;
- існує велика спільнота розробників C#, до якої ти можеш долучитися, щоб поставити питання, надати відповідь чи організувати мозковий штурм.

C# має блискучі перспективи завдяки популярності, універсальності та наявним програмним продуктам, створеним з використанням цієї мови[8].

2.2 Обґрунтування вибору середовищ розробки

Протягом багатьох років досягнення в області технологій і вимог призвели до різних способів написання додатків для Windows. Мета залишається незмінною: допомогти розробникам створити інтерфейс користувача і базовий стандартний код, який потім можна доповнити унікальними функціями. Ніхто не хоче писати код, який відображає текст пікселя за пікселем або малює контур меню або вікна.

Розробникам потрібен логічний, перевірений і надійний код, який надає всі ці функції (і багато іншого!).

Ось чому Microsoft створила багато інструментів і бібліотек (Microsoft Visual Studio ,UWP, WPF и Windows Forms).

Існує багато конструкторів баз даних з відкритим вихідним кодом, оптимізований спеціально для веб-додатків. Ці сервіси призначені для обробки великих обсягів даних і генерування численних паралельних запитів (Arduino IDE , IntelliJ IDEA , DataGrip, Eclipse).

Найпершою, а найчастіше, і єдиною програмою для початківців працювати з мікроконтролером стає Arduino IDE – інтегроване середовище розробки від творців платформи. У ній є все необхідне для розробки програм: написання коду, перевірка коду, компіляція, завантаження скетчу в мікроконтролер, монітор послідовного порту .

Також незаперечним плюсом є популярність даного середовища розробки, через що робота з ним полегшується в рази, адже в інтернеті є величезна маса уроків по налаштуванню і використанню даної середи розробки. Для Arduino IDE існує безліч плагінів для роботи з різними мікроконтролерами.

Середовище розробки Arduino IDE (рис.9) складається з вбудованого текстового редактора програмного коду, області повідомлень, вікна виведення тексту (консолі), панелі інструментів з кнопками часто використовуваних команд і декількох меню. Для завантаження програм і зв'язку середовище розробки підключається до апаратної частини Arduino.

Програма, написана в середовищі Arduino, називається скетч. Скетч пишеться в текстовому редакторі, що має інструменти вирізки/вставки, пошуку/заміни тексту. Під час збереження і експорту проекту в області повідомлень з'являються пояснення, також можуть відображатися помилки які виникли.

Вікно виведення тексту (консоль) показує повідомлення Arduino, що включають повні звіти про помилки та іншу інформацію. Кнопки панелі інструментів дозволяють перевірити і записати програму, створити, відкрити і зберегти скетч, відкрити моніторинг послідовної шини.

В Studio містяться інструменти для розробки рішень для смартфонів і планшетів, а також нові технологічні рішення для Android TV, Android Wear, Android Auto, Glass і додаткові контекстуальні модулі.

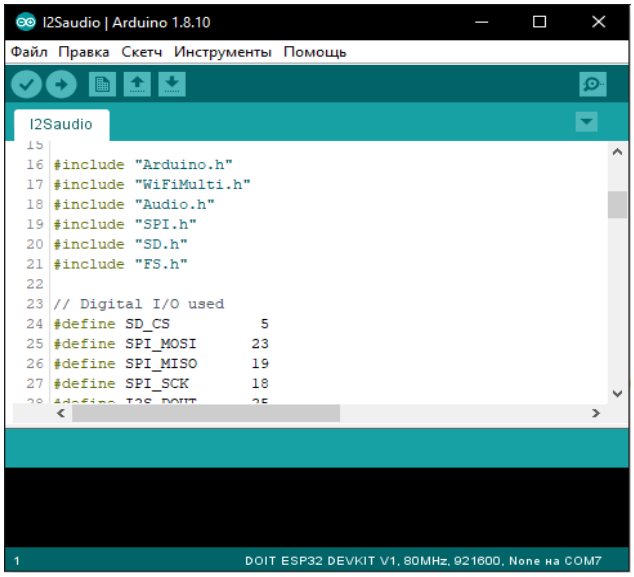

Рисунок 9 – Середовище розробки Arduino IDE

Рішення для Android розробляються в Android Studio з використанням Java або C++. В основі робочого процесу Android Studio закладений концепт безперервної інтеграції, що дозволяє відразу ж виявляти наявні проблеми.

Основні можливості IDE:

– інтелектуальний редактор з розширеним автодоповненням, рефакторингом і аналізом коду;
– функція "Миттєвий запуск", що дозволяє швидко перевіряти зміни, задавати параметри і запускати робочі цикли шляхом введення коду і зміни ресурсів, доступних з додатком, на пристрої або в емуляторі;
– швидкий і багатофункціональний емулятор Android з віртуальним акселерометром, датчиком робочої температури, магнітометром і іншими датчиками;
– підтримка всіх платформ Android: телефонів і планшетів, а також пристроїв Android Wear, Android Auto і Android TV;

- гнучка система збирання на основі Gradle з автоматизацією процесів формування коду додатків, управління залежностями і налаштованими конфігураціями файлів APK;
- шаблони коду для реалізації стандартних функцій;
- зручний редактор макетів з можливістю перетягування елементів і режимом прототипуваня для розробки додатків на інтуїтивному рівні.

Java Development Kit (JDK) є однією з трьох основних технологій, що використовуються в java-програмуванні. Вони також включають JVM (Віртуальна машина Java) і JRE (середовище виконання Java). Важливо розрізняти їх, а також розуміти, як вони пов'язані:

- JVM відповідає за виконання програм Java;
- JRE створює і управляє JVM;
- JDK дозволяє розробникам створювати програми, які можуть бути виконані і запущені через JVM і JRE.

Початківці розробники Java часто плутають JDK і JRE. Різниця полягає в тому, що JDK є інструментарієм розробки програмного забезпечення, тоді як JRE – це набір інструментів для запуску коду Java.

Усі середовища розробки які використовують Java, такі, як NetBeans, Sun Java Studio Creator, IntelliJ IDEA, Groovy, Eclipse, спираються на сервіси, що надаються JDK . Деякі з них для компіляції Java - програм використовують компілятор з комплекту JDK. Тому ці середовища розробки можуть включати в комплект постачання одну з версій JDK або вимагають для своєї роботи попередній інсталяції JDK на машині розробника. З певного часу фірма Sun надає повні вихідні тексти JDK, включаючи вихідні тексти самого Java-компілятора.

В теперішній час ринок пакетів програм для статистичної обробки даних пропонує велику кількість різноманітних Програмних забеспечень . Я пеглянув представлені на ринку різноманітні ПЗ .

Виходячи з поставленого мені завдання я обрав оптимальне і відповідне для нього ПЗ – статистичний пакет. Як правило, оптимальним є варіант, що комбінує в собі високий рівень продуктивності ПЗ, потрібні функціональні можливості і помірну ціну.

JetBrains – чеська компанія з розробки програмного забезпечення (рис.10). З 2000 року компанія випускає повнофункціональні, потужні і прості у використанні програмні продукти IntelliJ IDEA та DataGrip для розробки на мовах програмування Java, NET, Objective-C, Python, Ruby та інших.

Рисунок 10 – Логотип компанії JetBrains

DataGrip – це платформа для обробки елементів бази даних. У тому випадку, коли користувач займається таблицями і виконує в них різні дії, включаючи редагування стовпців, використання графічного інтерфейсу DataGrip буде дуже доречним. Такі корективи передбачають створення конкретного сценарію. Так, коригування вносяться безпосередньо в базу даних, або згенерований DDL-запит копіюється безпосередньо в редактор, а подальша робота ведеться згодом з самим кодом.

DataGrip (рис.11) орієнтований на підтримку автоматичного заповнення коду, що позитивно позначається на швидкості написання запитів. При введенні коду IDE розпізнає контекст і виконує звичні для користувача операції – він допомагає в написанні коду з урахуванням конкретних ключових слів і назв елементів бази даних.

Рисунок 11 – Логотип компанії DataGrip

Інструмент IDE розпізнає, які елементи бази даних використовуються в коді. Тому, якщо змінити ім'я одного з них в запиті, відповідні зміни відбудуться в базі даних. Існує функція пошуку, де елемент або символ використовувався в частині запиту. Крім того, є можливість переходу з використання в точку оновлення. Коли користувач виконує ті ж дії на об'єкт, вже згенерований в базі даних, курсор переміщує його у вікно структури бази даних. Коли запитується конкретне ім'я елемента, якого немає в базі даних (наприклад, ім'я таблиці або рядка вказано неправильно), IDE повідомляє про наявність помилки і дає рішення.

Редактор таблиць DataGrip здатний фільтрувати інформацію, а також текстову навігацію в таблиці. Крім того, якщо є зв'язок між зовнішніми «кей», можна загорнути в таблиці рядки, що посилаються на ці клавіші, і назад знову. Щоб створити частину запиту, слід позначити код і запустити його. У вікні результату запиту можна використовувати велику кількість параметрів редактора таблиць – з його допомогою можна коригувати дані, а також в ньому є текстова навігація.

Eclipse (рис. 12) – в першу чергу повноцінна Java IDE, націлена на групову розробку: середовище інтегроване з системами управління версіями – CVS, для інших систем (наприклад, Subversion, MS SourceSafe) існують плагіни. З огляду на безкоштовність та високу якість, Eclipse в багатьох організа-

ціях є корпоративним стандартом для розробки додатків. Також Eclipse – служить платформою для розробки нових розширень, тому він завоював популярність(будь-який розробник може розширити Eclipse своїми модулями). Вже існують C / C + + Development Tools (CDT), що розроблені інженерами QNX разом із IBM, і засоби для мов COBOL, FORTRAN, PHP та інші від різних розробників. Безліч розширень доповнює середу Eclipse менеджерами для роботи з базами даних, серверами додатків та інших. Eclipse написана на Java, тому є платформо-незалежним продуктом, крім бібліотеки графічного інтерфейсу SWT, який розробляється для всіх поширених платформ. Бібліотека SWT використовується замість стандартної для Java бібліотеки Swing. SWT повністю спирається на операційну систему, що забезпечує швидкість і звичний зовнішній вигляд інтерфейсу, та іноді викликає на різних платформах проблеми сумісності та стійкості додатків. Це середовище розробки зараз контрольована компанією Eclipse Foundation. Вона надає відкритий вихідний код, який забезпечує нові можливості для розробників.

Рисунок 12 – Логотип компанії Eclipse.

Даний проект – це інфраструктура, яка надає важливі для розробників базові сервіси. Наявні інструменти дозволяють створювати нові технології в Eclipse. Середовище розробки є не просто зборами API – вона може впоратися з повноцінними завданнями.Величезна кількість плагінів з відкритим вихідним кодом надає необмежені можливості для розробників інструментаріїв. В програму можуть бути додані будь-які доповнення, що в підсумку дозволить

налаштувати і адаптувати її під будь-яке завдання. Середовище програмування володіє наступними особливостями: Широка збірка API для додавання нових модулів, а також фреймворк для програмування будь-яких розширень. Підтримка всіх популярних операційних систем.

IntelliJ IDEA (рис.13) – це особливе середовище програмування або інтегроване середовище розробки (IDE), багато в чому призначене для Java. Це середовище використовується спеціально для розробки програм. Він розроблений JetBrains (раніше відомий як IntelliJ) і є доступний у двох виданнях: Community Edition, що має ліцензію Apache 2.0, і комерційне видання, відоме як Ultimate Edition. Обидва вони можуть бути використані для створення програмного забезпечення, яке можна продати.

Рисунок 13 – Логотип компанії IntelliJ IDEA.

Кожен компонент IntelliJ IDEA призначений для максимальної продуктивності розвитку. Розумний редактор коду в поєднанні з ергономічним дизайном роблять розробку не тільки ефективною, але і приємною.

Причини, по яким він вважає одним з найкращих інструментів програмування:

- легкий старт;
- функції допомоги;
- ергономічне середовище;
- глибоке розуміння коду;
- сполучення клавіш для будь-якої дії;
- стандартні та настроювані теми;
- багатомовний розвиток;

– завершення смарт-коду;

– продуктивність работи.

IntelliJ IDEA продуманий в кожному аспекті і готовий до використання відразу після установки. Навколишнє середовище забезпечує швидкий доступ до всіх функцій і вбудованих інструментів, які потрібні розробнику, а також широких варіантів налаштування. Є можливість повністю налаштувати середовище відповідно до робочого процесу: встановити комбінації клавіш, встановити плагіни, налаштувати інтерфейс, як вам подобається.

Після індексації вихідного коду IntelliJ IDEA надає безліч можливостей для швидкого та ефективного розвитку: розумне автозаповнення, аналіз коду в режимі реального часу та надійний рефакторинг.

Microsoft Visual Studio – лінійка продуктів компанії Microsoft (див.рис.14), що включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. IDE MS Visual Studio .NET – це набір інструментів і засобів розробки різного роду застосувань (консольних, Windows, мобільних, Web-застосувань) та сервісів. MS Visual Studio є мультипрограмним середовищем, що підтримує декілька мов програмування, зокрема, C++, C#.

Рисунок 14 – Логотип програмного забезпечення Visual Studio

Довідкові відомості про всі розробки компанії Microsoft зібрані в один програмний продукт MSDN (Microsoft Developer Network).У MS Visual Studio кожне окреме застосування є рішенням (solution), що складається з одного чи декількох проектів (project). Одночасно можна відкрити тільки одне рішення

(з розширенням. sln), при роботі над кількома рішеннями одночасно потрібно запускати декілька вікон Visual Studio[6].

Visual Studio надає шаблони для проектів найбільш поширених типів. Використання проектів і їх шаблонів дозволяє користувачеві зосередитися на реалізації окремої функції, в той час як проект буде виконувати загальне управління та завдання побудови.

Для створення нового проекту використовується майстер застосувань. Майстер створення програм надає користувальницький інтерфейс для створення проекту за шаблоном та створення шаблонів для файлів вихідних текстів. Майстер налаштовує структуру програми, основні меню і панелі інструментів, забезпечує включення деяких заголовних файлів.

Запуск середовища здійснюється через відповідний пункт меню Пуск (Пуск , Програми, MS Visual Studio). Після цього на екрані з'являється стартова сторінка (Start Page) MS Visual Studio (рис. 15).

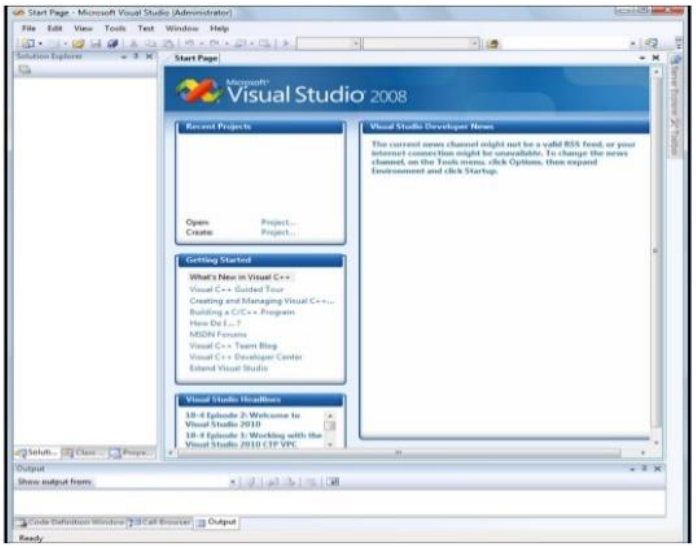

Рисунок 15 – Стартова сторінка MS Visual Studio 2008

Елементи графічного інтерфейсу MS Visual Studio характерні для Windows-програм. Розмір і форма вікон визначається конкретною конфігурацією системи. Користувач має можливість змінювати розмір і розташування

окремих елементів, згортати їх з тим, щоб збільшити місце для інших, необхідних в даний момент, елементів [7].

У головному вікні Visual Studio можна виділити декілька основних елементів (рис. 16):

- меню та набір інструментальних панелей, де зосереджені команди для роботи в IDE;
- вікно Провідник рішень (Solution Explorer), що дозволяє переглядати склад проектів, що входять у рішення, у вигляді ієрархічної структури, а також зв'язки між проектами та їх компонентами;
- вікно редактора, що служить для набору тексту програми і підтримує автодоповнення та підсвітку ситнаксису;
- вікно виведення стану Output, в якому відображається інформація про хід побудови (збірки) програми та виявлені помилки.

Розмір цих зон та їх розташування на екрані користувач може налагоджувати за власним бажанням.

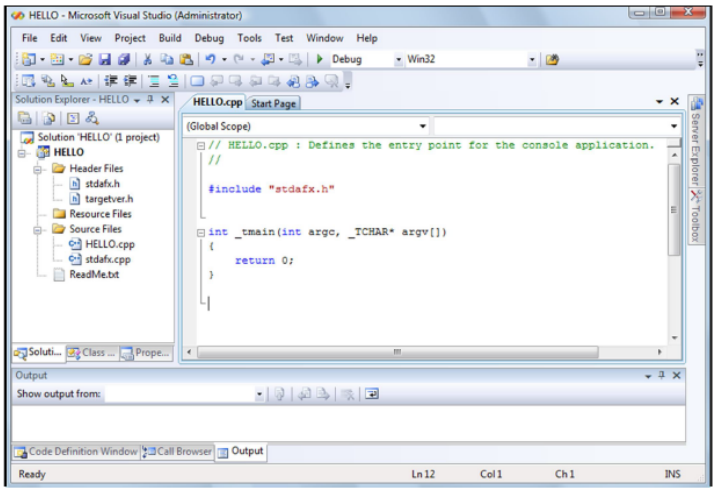

Рисунок 16 – Головне вікно Visual Studio 2008

Для створення проекта потрібно вибрати відповідний пункт меню середовища (File4New4 Project) або натиснути комбінацію клавіш Ctrl+Shift+N.

При цьому зявляється діалогове вікно New Project, яке дозволяє створювати різні типи проектів (рис. 17).

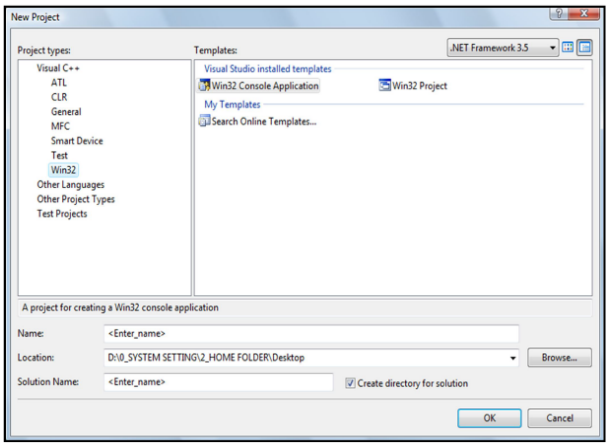

Рисунок 17 – Вікно New Project

Щоб створити консольного застосування[4], слід у розділі Project Types (Тип проекту) вказати значення Win32, а у розділі Templates (Шаблон) – значення Win32 Console Application (Консольний додаток). Після подальшого вказання імені проекта, його місцезнаходження і натиснення кнопки Ok здійснюється активізація майстра створення застосування (Application Wizard), який дозволяє задати потрібні параметри застосування (рис.18).

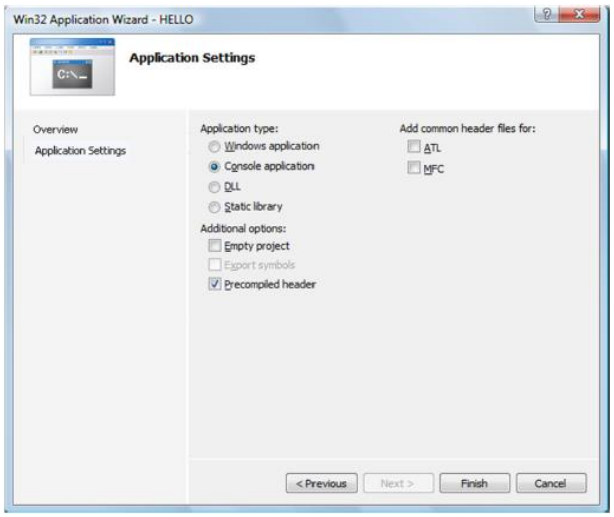

Рисунок 18 – Вікно майстра Application Wizard

Створення проекту призводить до створення на диску проекта вказаного типу, а також створюється рішення з вказаною назвою (разом з посиланнями на проект).

Переглянути вміст проекту, який зображено деревом файлів, які його утворюють, можна у вікні Solution Explorer (рис.19).

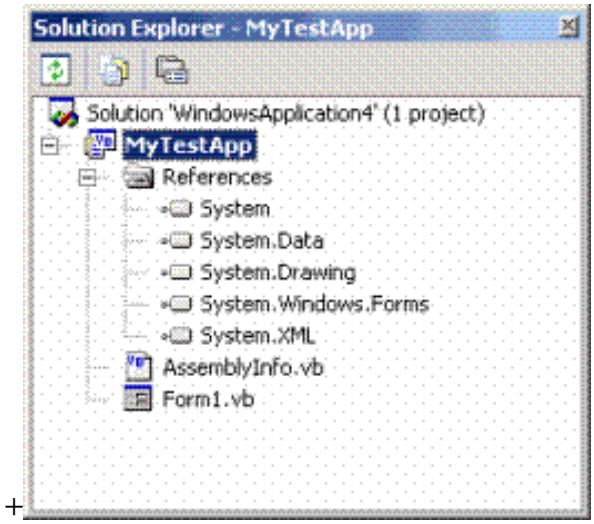

Рисунок 19 – Вікно Solution Explorer

Після створення порожнього проекту слід додати до нього принаймні вихідний файл (.cpp), який буде містити програмний код (наприклад, Project 4 New Item).

Visual Studio має спеціальні комбінації клавіш майже для всього, від перегляду останніх файлів до запуску та налагодження проекту. Однією з універсальних комбінацій є подвійний Shift (Пошук скрізь). Ця функція дозволяє знаходити будь-які об'єкти в проекті або за його межами. Діапазон пошуку може включати в себе все, від файлів, дій, класів і символів до налаштувань, елементів інтерфейсу та історії від Git.

Можливості Visual Studio :

– співпрацювати над проектами в режимі реального часу;

- автоматичне оновлення програмного забезпечення з офіційного сайту;
- створення додатків для Windows, Linux, Mac OS;
- використання розділених вікон XML і CSS;
- комплексне налагодження та статичний аналіз вихідного коду;
- мови програмування C++, C#, Visual Basic, Python, TypeScript, JavaScript, AJAX тощо.

Переваги Visual Studio:

- підтримка декількох моніторів;
- низькі системні вимоги;
- вбудований налагоджувач для JavaScript;
- можливість зв'язатися з фахівцями Microsoft;
- наявність користувальницького інтерфейсу російською мовою;
- інтегрована система відстеження та виправлення помилок;
- розробка графічних додатків з підтримкою технології Windows Forms;
- великий набір інструментів для написання якісного програмного коду.

Недоліки Visual Studio – не працює з Windows XP і Vista.

3. РЕАЛІЗАЦІЯ ТЕХНІЧНОГО ПРОЕКТУ

3.1 Розробка програмної системи

Для виконання Кваліфікаційної роботи (КР) на тему «Розробка десктопного додатку пошуку викладачів для додатковий заннять» розглянуто два середовища – середовище розробки IntelliJ IDEA для розробки на мові програмування Java та Visual Studio і мову програмування C#. При роботі на IntelliJ IDEA на Java виникла проблема при підключенні бази даних. Тому звернувся до самої популярної Visual Studio та мові програмування C#.

Для покращення інтерфейсу додав конструктор MaterialSkin.

Список справ які виконає MaterialSkin:

- панель прогресу – анімація та варіанти, можливо кругле завантаження;
- фон (можливо);
- деякі покращення кольорового коду та рефакторинг;
- наявність DatePicker;
- ящик – заголовок, роздільник, субтитри та прокрутка.

3.2 Підключення бази даних (СКБД) SQL Server

База даних може бути визначена як збір великої кількості даних, які зберігаються в комп'ютері в електронному вигляді. SQL Server надає нам платформу, яка зазвичай використовується для зберігання даних про ту саму сутність, де ми можемо оновлювати та керувати даними відповідно до наших вимог.

SQL Server в даний час є однією з найпопулярніших систем управління базами даних (СКБД) у світі. Вона підходить для самих різноманітних проектів: як для створення невеликих додатків так і для великих проектів з високим навантаженням. SQL Server створено корпорацією Майкрософт. Перша

версія була випущена в 1987 році. А поточна версія, яка вийшла в 2019 році і яка буде використовуватися в ході роботи.

QL Server вже давно є виключно системою управління базами даних для Windows, але зараз ця система також доступна на Linux.

SQL Server характеризується такими особливостями, як:

- швидкодія та продуктивність. SQL Server дуже швидкий;
- безпека та надійність. SQL Server забезпечує шифрування даних;
- локонічність та простота. З цією СКБД відносно легко працювати та адмініструвати.

Головним аспектом MS SQL Server, є база даних. База даних – це сховище даних, організоване з певними умовами .База даних зазвичай представляє файл на жорсткому диску, але таке зіставлення не завжди потрібно. Система керування базами даних або СКБД використовуються для зберігання та адміністрування баз даних. І просто MS SQL Server є однією з таких СКБД.

MS SQL Server використовує реляційну модель для організації баз даних. Ця модель бази даних була створена в 1970 році Едгаром Коддом. І зараз вважається стандартом організації баз даних.

Реляційна модель призначена для зберігання даних у вигляді таблиць, які складаються з рядків і стовпців. Кожен рядок зберігає окремий об'єкт, в стовпці розміщують атрибути цього об'єкта.

Первинний ключ потрібен для ідентифікації кожного рядка в таблиці. Первинним ключем може бути один або кілька стовпців. Посилатися на конкретний рядок в таблиці можно за допомогою первинного ключа. Це означає, що два рядки не можуть мати однаковий первинний ключ.

Одна таблиця може бути пов'язана з іншою за допомогою клавіш, тобто зв'язки можуть бути організовані між двома таблицями.

Для взаємодії з базою даних використовується мова SQL (мова структурованих запитів). Клієнт відправляє запит в SQL за допомогою спеціального API. СКБД правильно інтерпретує і виконує запит, а потім відправляє результат виконання клієнту.

MS SQL Server має різні варіанти. Існує MS SQL Server Developer Edition. Це повнофункціональний реліз, що містить весь функціонал, і призначений тільки для потреб в розробці. Однак ця версія не може використовувтись для реальних проектів, для розгортання в якості реального сервера. А для вивчення всієї механіки MS SQL Server ця версія є оптимальним варіантом, тому в роботі ми будемо використовувати цю версію.

Потрібно налаштувати SQL Server в операційній системі Windows з метою його використання. Нижче наведені кроки, які треба виконати, щоб завершити установку в нашій системі.

Встановлення SQL Server в ОС проводилось етапами :

Крок 1: Перейшли до https://www.microsoft.com/en-gb/sql-server/sql-server-downloads.

Крок 2. Завантажили версію розробника.

Крок 3: Після завантаження налаштування відкрили його, натиснувши двічі та вибрали «Завантажити медіа».

Крок 4: Вибераємо файл ISO, шлях та натисніть «Завантажити».

Крок 5: Витягнули завантажений файл і відкрили папку.

Крок 6: Клацнули на setup.exe і натиснули на перше посилання.

Крок 7: Кілька разів натиснули наступну кнопку.

Крок 8:Прийняли цей термін і вибрали встановити єдиний варіант.

Крок 9: Почекали 10-15 хвилин, щоб встановити його.

Крок10: Перейшли на https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-2017.

Крок 11: Завантажили студію управління SQL-сервером та запустили налаштування.

Крок 12: Після його встановлення перезавантажили систему, і закінчили.

За допомогою цих простих кроків ми завантажили та налаштували SQL в нашій операційній системі. Після його встановлення ми використовуємо його для зберігання та обробки даних.

Після установки відкриваємо студію управління SQL Server, щоб розпочати роботу. Студія надає нам платформу взаємодії для роботи з базою даних. Тому для роботи з SQL-сервером ми використовуємо мову під назвою SQL (Структурована мова запитів).

Коли ми вперше відкриємо цю студію управління, нам довелося вибрати сервер, на якому ми будемо працювати.

3.3 Вибір технології для реалізації додатку .

Метою даної роботи є створення додатку пошуку викладачів для додаткових знань в середовищі Visual Studio на мові програмування C#.

Суть проекту полягає в тому, щоб створити простий редактор списків викладачів з можливістю збереження даних у файл і відновлення їх з файлу. Полями записів, що відповідають викладачу , будуть його повне ім'я та прізвище, предмет, який він викладає , вартість заняття .

Отже, для створення додатку на C# по-перше, нам потрібен текстовий редактор, в якому ми можемо роздрукувати код програми. По-друге, нам потрібен компілятор, який би компілював код, введений в текстовому редакторі, в exe-додаток. По-третє, нам потрібен фреймворк .NET, який необхідний для складання та виконання програми.

Для створення додатків в C# будемо використовувати безкоштовне і повнофункціональне середовище розробки – Visual Studio Community 2022, яке можна завантажити за адресою: Microsoft Visual Studio 2022.

Щоб додати підтримку проекту для C# і .NET 6 до Visual Studio, можна вибрати лише ASP.NET та розробку веб-застосунків у програмі інсталяції. Є можливість вибрати більше варіантів і навіть всі варіанти, але треба врахувати вільний розмір на жорсткому диску – чим більше варіантів вибрано, відповідно, тим більше місця на диску буде зайнято.

При установці Visual Studio (рис.20) на вашому комп'ютері будуть встановлені всі необхідні інструменти для розробки програм, в тому числі фреймворк .NET 6.

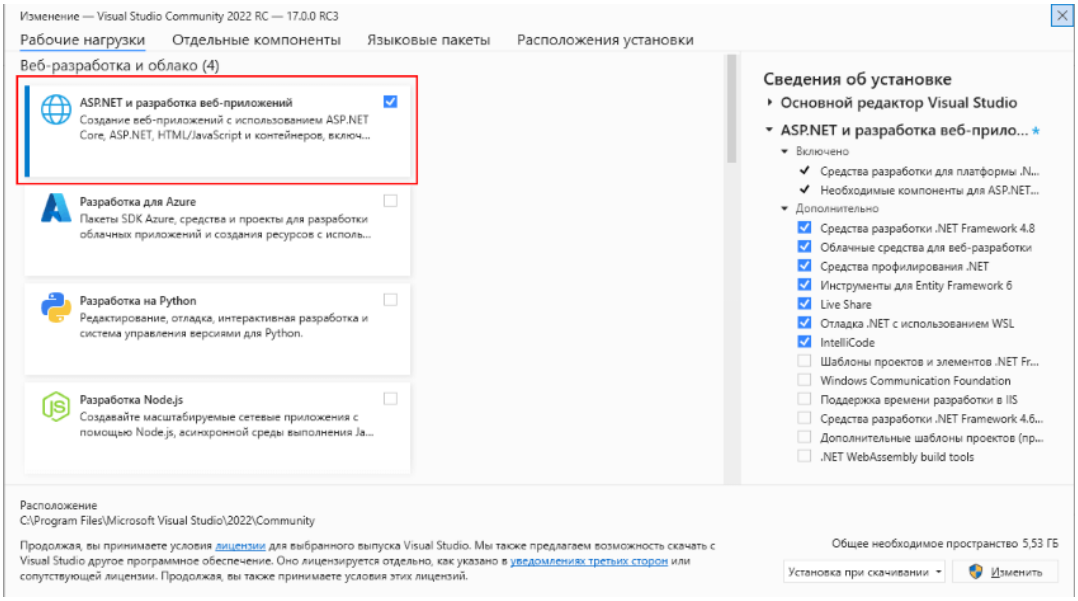

Рисунок 20 – Запуск програми та початкова конфігурація програми

При створенні програм у Visual Studio можна використовувати два підходи. Перший передбачає створення окремих проектів для кожної задачі. Алгоритм роботи наступний (рис. 21):

1. Запускаємо Microsoft Visual Studio.
2. На початковій сторінці вибираємо Создать проект – з’являється вікно Создание проекта.
3. В верхній частині вікна вибираємо Visual C# --> Windows --> Приложение Windows Forms (для створення віконного додатку) або Консольное приложение (для створення консольної програми).
4. В нижній частині вікна вводимо ім’я проекту (поле Имя:); за допомогою кнопки Огляд вказуємо місце збереження проекту (поле Расположение:); перевіряємо наявність вибору Создать каталог для решения.
5. Натискаємо OK.

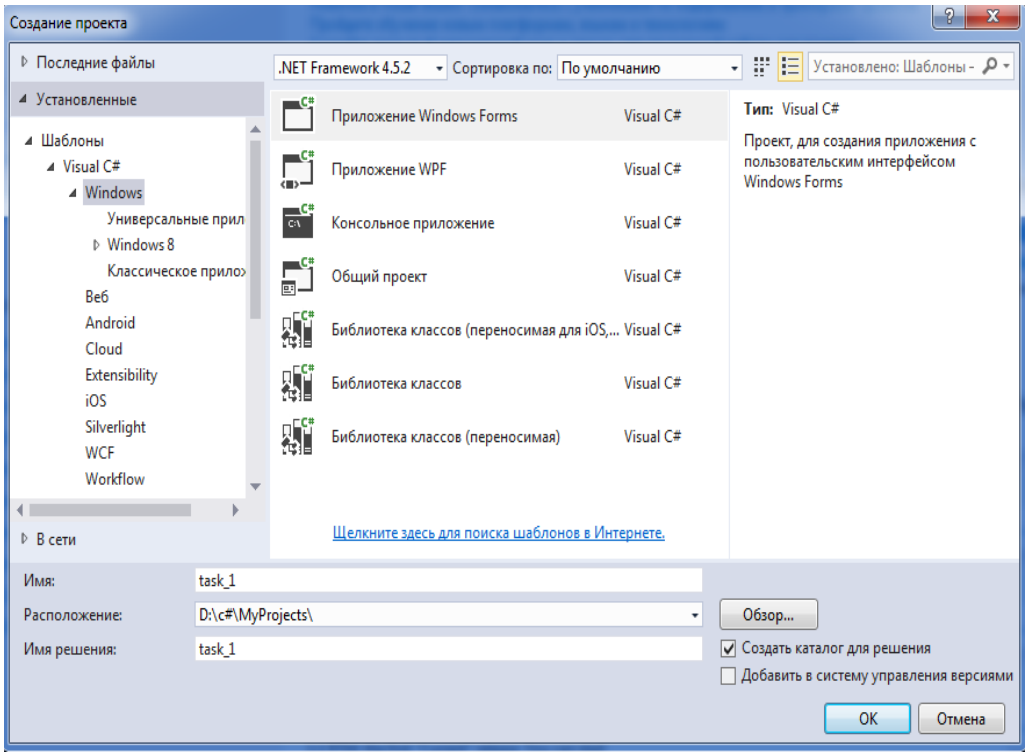

Рисунок 21 – Вибір версії

Наступне вікно Visual Studio запропонує нам вибрати версію. NET, яка буде використовуватися для проекту. За замовчуванням тут вибрана остання версія – .NET 6.0. Залишимо і натисніть кнопку Create (Створити) для створення проекту.

Після цього Visual Studio створить та відкриє нам проект.

3.4 Структура програми

У великому полі в центрі, яке по суті є текстовим редактором, знаходиться згенерований за замовчуванням код C#. Згодом ми змінимо його на свій (рис.22)

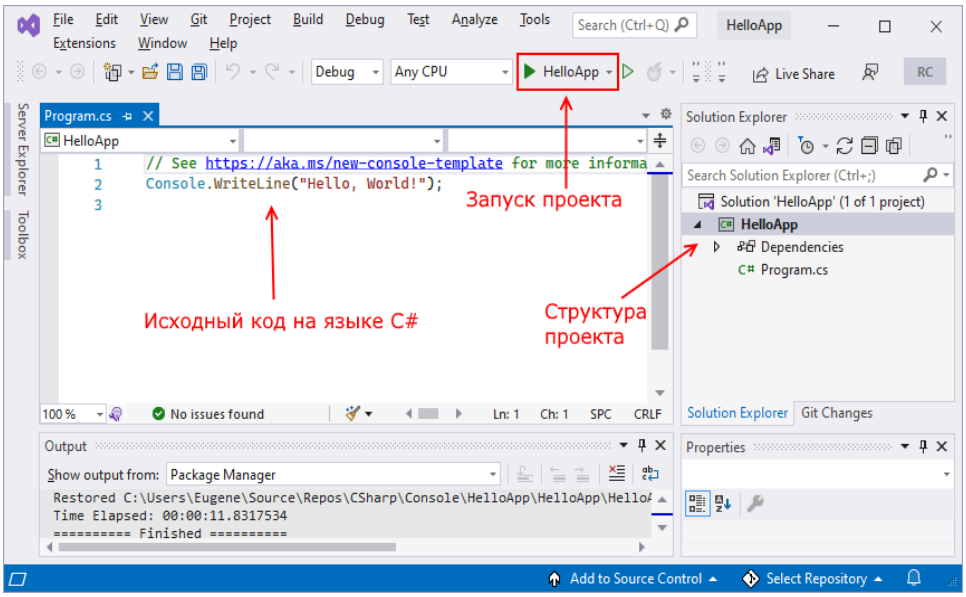

Рисунок 22 – Структура пректу

Вікно Solution Explorer знаходиться справа, в ньому можна побачити структуру нашого проекту. В даному випадку згенерована за замовчуванням структура така : вузол Dependencies – це вузол містить складання dll, що додані до проекту за замовчуванням. Ці збірки містять класи бібліотеки NET, що використовуватиме C#.Якщо збірка непотрібні потім її можна видалити, а якщо знадобиться то додати потрібну бібліотеку.

Далі бачимо сам файл коду програми Program.cs (рис. 23), який за замовчуванням відкритий у центральному вікні і має лише два рядки:

```
1  // See https://aka.ms/new-console-template for more information
2  Console.WriteLine("Hello, World!");
```

Рисунок 23 – Console.WriteLine("Hello, World!");

У першому рядокку ми бачимо символи // які представляють коментарі – пояснення до коду.

А другий рядок є кодом програми: Console.WriteLine(«Hello World!»);. Цей рядок виводить на консоль рядок «Hello World!».

Хоча програма містить лише один рядок коду, це вже деяка програма, яку можемо запустити. Запускаємо проект за допомогою клавіші F5 або з панелі інструментів, натиснувши на зелену стрілку. Якщо все зробили правильно, то при запуску програми на консоль буде виведено рядок«Hello World!».

Змінній name надається результат методу Console.ReadLine(), що дозволяє завжди рахувати з консолі введений рядок. Так вводячі в консолі рядок (точніше ім'я) цей рядок опиниться у змінній name.

Для того щоб ввести значення змінної name всередину рядка, яка виводиться на консоль, застосовують фігурні дужки {}. Так при виведенні рядка на консоль вираз {name} замінюватиметься на значення змінної name – (введене ім'я).

Але, щоб можна було вводити таким чином значення змінних всередину рядка, перед рядком вказують знак долара $.

Проект протестуємо та запустивши його на виконання, натиснувши F5 або зелену стрілочку. Важливою частиною роботи є тестування та документування створюваного продукту. Цей етап завершується створенням дистрибутиву програми чи його частин і документуванням процедури його інсталяції.

Коли проект створено, знайти його можна на жорсткому диску в папці проекту в каталозі bin\Debug\net6.0. Воно буде називатися на ім'я проекту і матиме розширення exe. Потім цей файл запускається без Visual Studio, а також переноситься на інші комп'ютери, де встановлено NET 6.

Таким чином завершивши роботу програми ми отримали каркас програми з головним меню, панеллю інструментів, рядком стану, можливістю зберігати та відновлювати документ із файлу.

В кінці роботи роботи майстра визначили базовий клас для класу виду програми. У нашому випадку підійде CFormView, який дозволить організувати клієнтську область вікна програми на основі діалогової панелі, яку легко

можна змінити редактором ресурсів. У нас це діалогова панель з ідентифікатором IDD_MFC_FORM, на якій розташований єдиний елемент управління ListBox для виведення даних на екран.

Після введення коду програму можна компілювати, скориставшись комбінацією клавіш Ctrl+F7 або командою меню Build Compile. Коли компілятор виявить синтаксичну помилку, то виведе відповідне повідомлення у вікно Output.

Компіляцію разом із компонуванням виконали за допомогою команд меню Build, Build Solution і Build,Build . У результаті успішного компонування програми створюємо exe-файл.

Після перевірки програму, в якій виправлено синтаксичні помилки, запускаємо на виконання клавішею F5 або комбінацією клавіш Ctrl+F5. Результати роботи програми ми можемо побачити у консольному вікні.

Якщо виникає помилка, то робота програми переривається і виводиться повідомлення про помилку часу виконання (run-time error). Програма може «зациклитися» – в такому разі постає потреба у примусовому перериванні її виконання. Для примусового зупинення програми достатньо просто закрити консольне вікно.Для продовження редагування програми, що була раніше записана до файла, слід відкрити проект чи рішення командою File Open Project/Solution або натисканням комбінації клавіш Ctrl+Shift+O.

Налагодження – це процес пошуку і виправлення помилок в програмі, що перешкоджають коректній роботі. Налагодження програми є найбільш важливим і трудомістким етапом її розробки.

Налагодження програм передбачає виконання наступних кроків:

- виявлення факту наявності помилки;
- визначення місця знаходження помилки (її локалізація);
- виправлення помилки.

Загалом існує три основних типи помилок:

- синтаксичні, що виникають в результаті порушення правил написання алгоритмічних конструкцій (речень) мови програмування;

- семантичні, зв'язані з недопустимими значеннями параметрів, діями над параметрами (ділення на 0, спроба відкрити неіснуючий файл тощо);
- логічні, зв'язані з неправильним використанням алгоритмічних конструкцій, неправильним проектуванням і реалізацією програми.

Синтаксичні помилки виявляються на етапі компіляції програми, тому їх ще називають помилками етапу компіляції. Ці помилки відображаються у вікні вихідних даних Output (номер рядка, в якому знайдена помилка, і її опис). Натисення кнопки Go To Next Message (зелена стрілка праворуч у вікні Output) або подвійне клацання лівою кнопкою миші на повідомленні про помилку, переводить курсор на рядок, що її містить.

Семантичні помилки виявляються під час роботи програми, тому їх ще називають помилками етапу виконання. При виявленні такої помилки, виконання програми завершується і виводиться вікно з повідомленням про тип помилки і адресою інструкції, в якій вона сталася. Наприклад, логічні помилки не викликають порушень в роботі програми, але приводять до неправильних результатів (некоректному або непередбачуваному значенню змінних, невиконанню коду, коли це передбачається, тощо). Ці помилки часто важко відслідкувати, оскільки IDE не може знайти їх автоматично, як синтаксичні і семантичні.

Логічні помилки, зазвичай, виявляються при тестуванні програми з використанням вбудованого налагоджувача, функції якого реалізуються командами меню Debug.

Програму у покроковому режимі запускаємо командою Debug, Step Into (клавіша F11) або Debug Step Over (клавіша F10). Команда Step Into застосовується для трасуванням (покрокового виконання) програми із заходом всередину функцій, тоді як команда Step Over виконує функцію за один крок.

Для завершення покрокового виконання програми використовуємо команду Debug, Stop Debugging або комбінація клавіш Shift+F5.

Може допомогти знайти помилки в алгоритмі програми трасування програми, але зазвичай бажано також знати, що відбувається на кожному кроці зі значеннями окремих змінних. Непрогнозовані значення означають що є помилки.

Для спостереженням за поточними значеннями програмних об'єктів у вбудованому налагоджувачі є засоби перегляду значень змінних, виразів і структур даних.Щоб дізнатися значення змінної, треба затримати на ній покажчик миші. При цьому поряд з іменем змінної на екрані з'являється підказка із значенням цієї змінної.

Окрім екранної підказки, змінні із своїм поточним значенням відображаються у вікнах WatchX, які активізуються командою Debug, Windows, Watch, WatchX, де X – порядковий номер вікна. Для перегляду значення певної змінної її ім'я слід вказати у вікні Watch в колонці Name, її поточні значення виводитимуться у колонці Value . Коли вікно Watch активне, додати до нього змінну можна як у звичайну таблицю, а для видалення змінної слід обрати її ім'я і натиснути клавішу Delete.

Організація точок переривання. Для детального відстеження дій програми в проблемних місцях, у відповідних рядках програми встановлюють точки переривання (breakpoints). В час виходу програми на такі точки її виконання припиняється. При цьому можна переглянути значення деяких елементів даних, продовжити виконання збійної ділянки програми в покроковому режимі і т.ін.

Щоб задати точку переривання перед деяким оператором, необхідно встановити перед ним курсор і вибрати команду Debug Toggle Breakpoint (клавіша F9) або клацнути мишею на сірому полі ліворуч від вибраного оператора. Точка переривання позначається червоним колом на лівому полі вікна текстового редактора .Повторна дія (клацання мишею на точці переривання або натиснення F9) знімає (видаляє) точку переривання. Видалити усі точки переривання можна командою Debug Delete All Breakpoints.

Програма запускається в налагоджувальному режимі за допомогою команди Debug Start Debugging (клавіша F5). В результаті код програми виконується до рядка, на якому встановлена точка переривання.

Після зупинки програми можна розпочати її покрокове виконання для аналізу поточних значень змінних в процесі пошуку логічної помилки. Значення змінних на кожному кроці виконання програми можна побачити у спливаючих підказках, які з'являються при наведенні курсору на змінну, або у вікні змінних (Watch).

Подальше продовження роботи програми до наступної точки переривання здійснюється повторним її запуском (F5).

Створення умовних точок переривання. IDE підтримує також створення додаткових умов для спрацьовування точок переривання. Зокрема, в точках переривання можна задати декілька додаткових параметрів:

- Location – місце зупинки (рядок і модуль);
- Condition-перевірка умов зміни (на рівність якому-небудь значенню або на зміну значення);
- Hit Count-перевірка на лічильник проходів;
- Filter-комбінація декількох умов для значень змінної;
- When Hit – задає дії при виникненні зупинки.

Для цього треба активізувати контекстне меню над рядком з точкою переривання і вибрати команду Breakpoint.

Рекомендації щодо налагодження програм:

- не слід розміщувати в одному рядку програми більше одного оператора;
- програмувати треба з поступовим нарощуванням (за можливості кодувати і налагоджувати програму невеликими частинами);
- треба уникати побудови функцій, розмір яких більше 25-30 рядків (у противному випадку слід розбивати їх на декілька менших за розміром функцій).

Запис – це структура даних, що складається з фіксованої кількості взаємопов'язаних між собою компонентів одного або декількох типів. Компоненти запису називають полями. Кожне поле характеризується своїм ім'ям і типом значень. Записи найчастіше використовуються при створенні різного роду інформаційних систем, які містять різнорідні відомості. Наприклад, у інформаційній системі про успішність студентів запис може містити в собі ПІБ студента, курс, групу, оцінки по окремих предметах в кожну екзаменаційну сесію, наявність і кількість перездач по кожному предмету, а якщо необхідно, то і дати іспитів. Частина даної інформації представляється цілими числами, для деяких відомостей краще підходить представлення їх у вигляді літерних рядків, а деякі дані можуть мати логічний тип. Як видно, в такому структурному типі представлені компоненти з абсолютно різними типами значень.

Анкети репетиторів можна переглянути через пошукову панель в розділі «Викладачі» (рис. 26). Викладача треба шукати самостійно. Для цього є чималий список критеріїв у меню. Слід обрати параметри фільтрування: предмет, індивідуальні заняття, або групові, а також максимальну ціну, що задовільнять учня.

Для входу в додаток необхідно зареєструватися. Для цього необхідно придумати логін та пароль. Також потрібно вибрати, ким Ви являєтеся: учнем або викладачем (рис. 24).

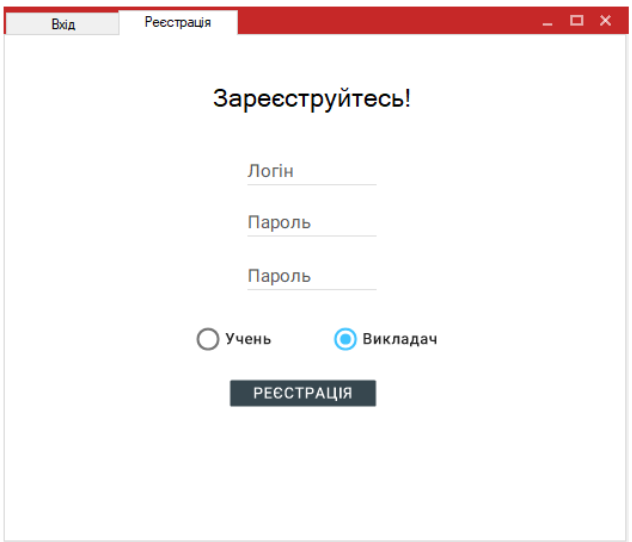

Рисунок 24 – Вікно реєстрації

Після реєстрації Вам потрібно повторно ввести Ваш логін та пароль у вкладці «Вхід» для того, щоб увійти в обліковий запис (рис. 25).

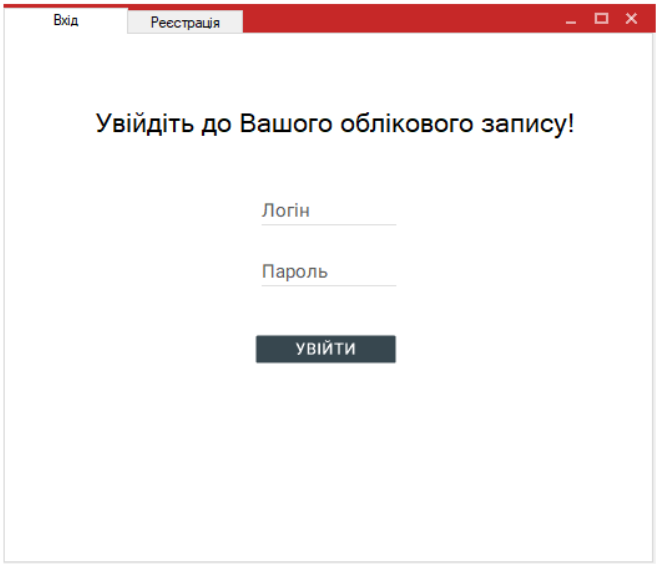

Рисунок 25 – Вікно для входу в обліковий запис

Після входу в обліковий запис Ви побачите анкети вчителів (рис. 26). Саме тут Ви можете порівняти викладачів за цінами та відгуками.

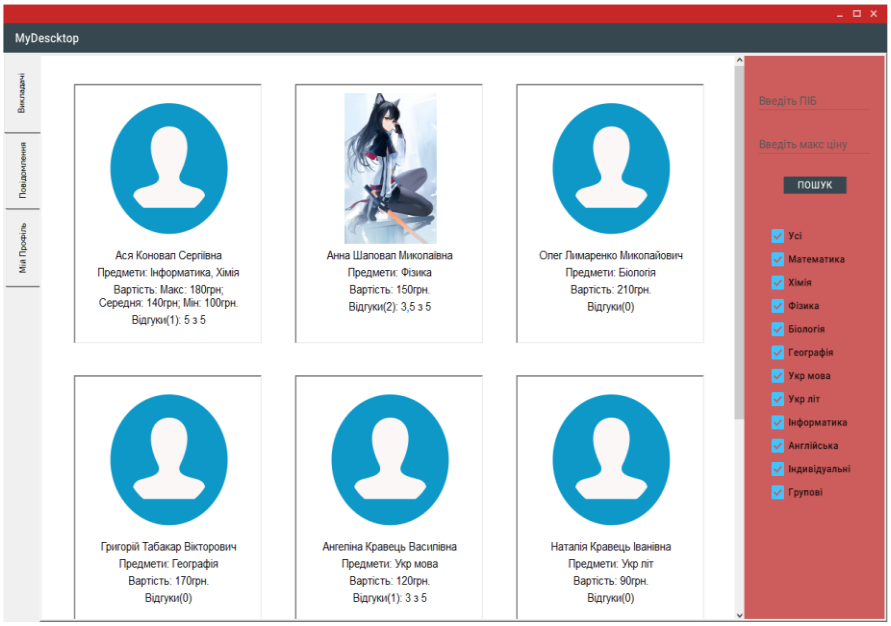

Рисунок 26 – Вкладка з анкетами викладачів

При подвійному натисканні на анкету вчителя у Вас з'явиться його профіль (рис. 27). У профілі Ви можете детальніше ознайомитись з усіма предметами, які викладає даний вчитель, з цінами та описом до кожного предмета. Тут же Ви можете залишити відгук про викладача та написати йому повідомлення.

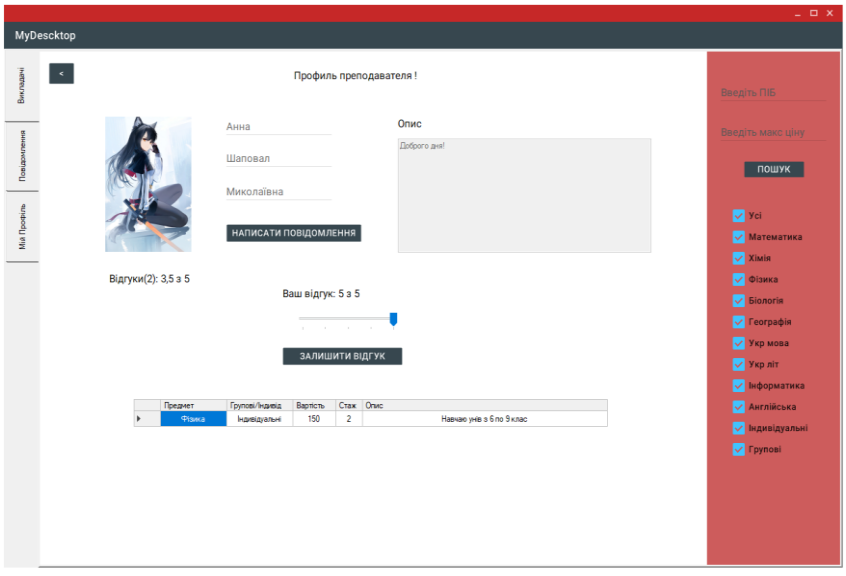

Рисунок 27 – Профіль анкети викладача

Для створення анкети, якщо Ви являєтесь вчителем, Вам необхідно буде заповнити всі поля з даними на вкладці «Мій Профіль» (рис. 28). Після заповнення усіх даних, анкета автоматично буде створена та відобразиться на вкладці «Викладачі». Вашу анкету зможуть побачити інші користувачі додатка.

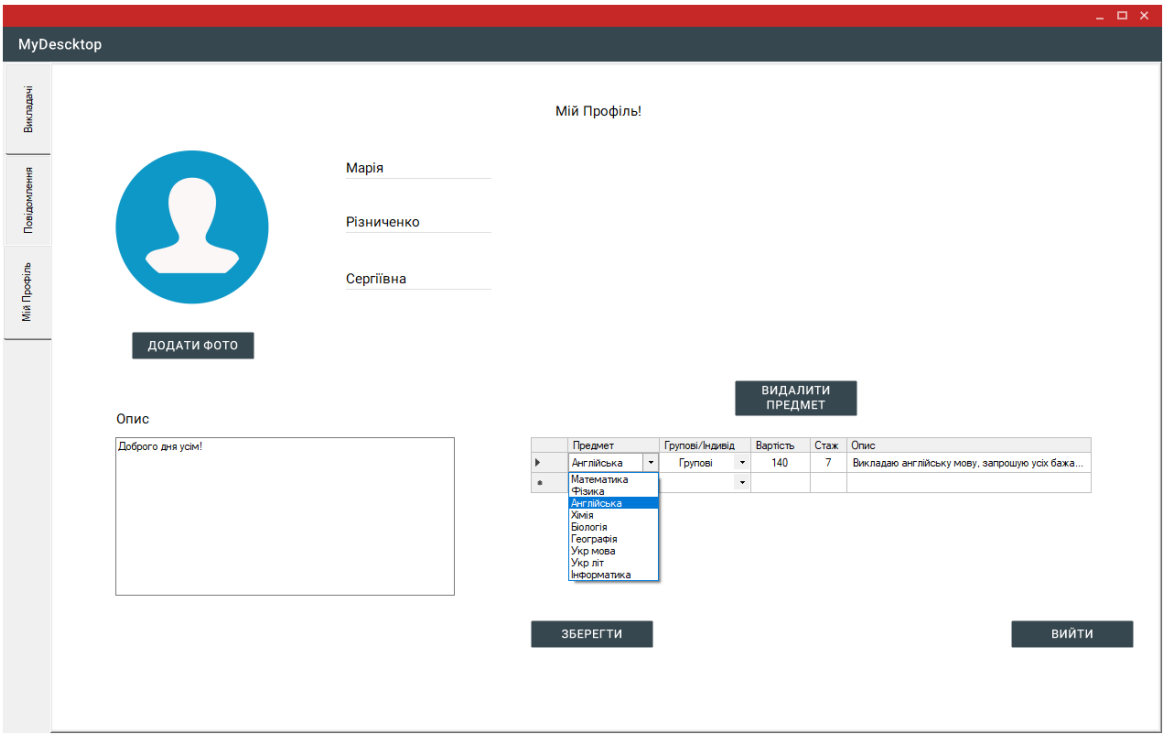

Рисунок 28 – Вкладка для заповнення профіля викладача

Якщо певний вчитель відповідає Вашим вимогам, необхідно повністю заповнити всі дані у вкладці «Мій Профіль» (рис. 29). Після цього Ви зможете надіслати заявку обраному вчителю, або написати повідомлення викладачу для уточнення деталей. На Ваше повідомлення, або заявку відповість сам викладач. Після того, як обрали викладача, учень може написати повідомлення викладачу для уточнення деталей, або, щоб залишити заявку. На Ваше повідомлення, або заявку відповість сам викладач

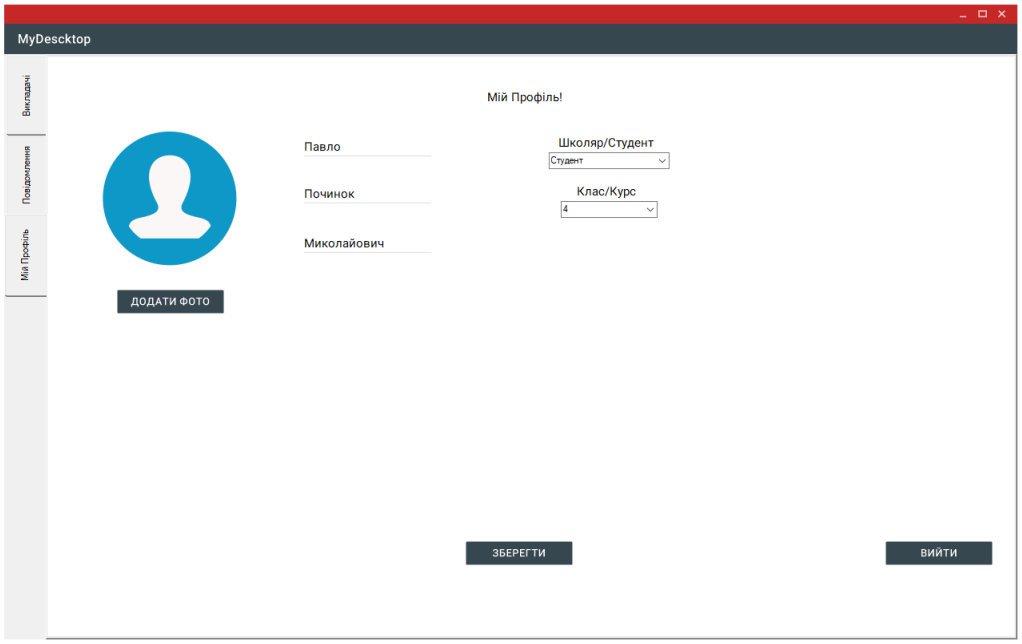

Рисунок 29 – Вкладка для заповнення профіля учня

Для швидкого пошуку викладача на вкладці «Викладачі» також є пошукова панель (рис. 30).

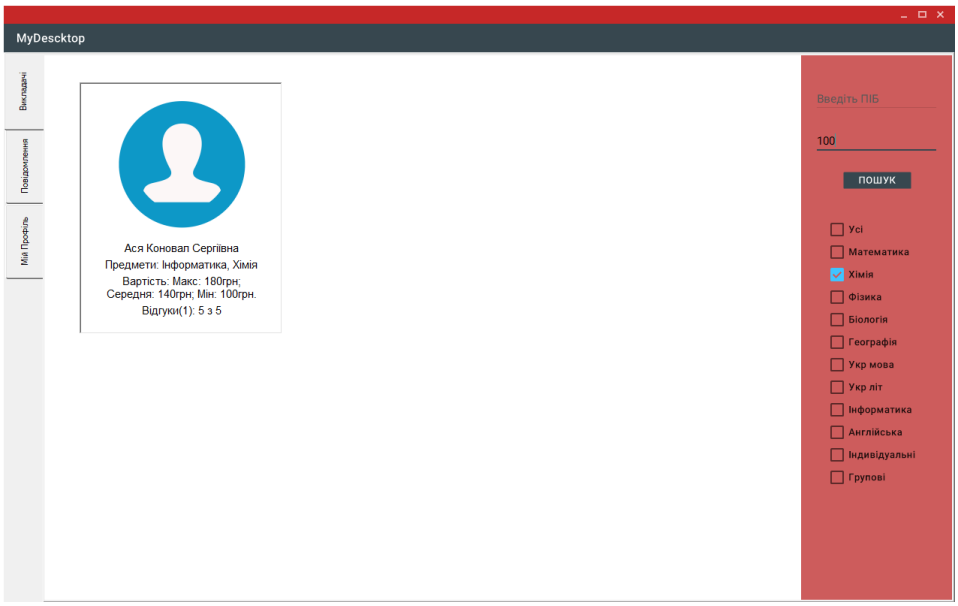

Рисунок 30 – Приклад роботи пошукової системи

Панель пошуку містить два параметри: ФІО викладача та максимальна ціна за одну годину. За ними впорядковані всі предмети викладачів. Ви можете відібрати викладачів за цікавим для вас предметом, поставивши галочку біля назви предмета на панелі пошуку (рис. 30). Можна обрати не тільки індивідуальні заняття, а й навчання в невеликих групах. Деякі викладачі пропонують, крім уроків, разові консультації з певних тем або підготовку до контрольних робіт. Все це можна буде побачити в описі профіля викладача .

Після того, як обрали викладача, учень може написати повідомлення викладачу для уточнення деталей, або, щоб залишити заявку. На Ваше повідомлення, або заявку відповість сам викладач.

3.5 Інсталяційне тестування додатку.

- успішність запуску програми після установки;
- розташування програми в файловій системі за замовчуванням;
- розташування програми в файловій системі, якщо шлях збереження змінений користувачем;
- наявність ярликів на робочому столі, узагалі можливість туди його імпортувати;
- чи є встановлений компонент в меню Пуск ➞ Програми;
- установка програми для поточного користувача / для всіх користувачів комп’ютера;
- видалення програми різними способами: на Windows запускаємо uninstall.exe, просто натискаємо кнопку видалення або шляхом видалення папок.

ВИСНОВОК

Результатом Кваліфікаційної роботи є створений продукт – десктопний програмний додаток , реалізований на мові програмування C#, з використанням Visual Studio за допомогою якого є можливість проводити пошук викладачів для додатковий знань, виправляти текстовий файл, поповнювати інформаційну базу даних викладачів.

Під час виконання цього проекту розроблено інформаційну базу даних викладачів, яка допоможе будь-якому користувачеві легко знайти потрібну інформацію з будь-якого предмета та знайти викладача для додаткових занять.

У додатку можна буде залишити відгуки про кваліфікацію викладача, що дозволить швидко знайти хорошого вчителя.

Дана кваліфікаційна робота особлива тим , що в ній дитина може знайти безкоштовного репетитора між однолітків. Це простір , який об’єднуватиме дітей-волонтерів із дітьми з України.

Додаток дозволяє записатися на заняття до школяра-волонтера в пару кліків. Заняття проходить в онлайн форматі і для нього потрібні лише Google та ноутбук/комп’ютер.

З проектом вже ознайомлені декілька школярів-волонтерів і вони готові зараз допомагати українським дітям з адаптацією в стресовій ситуації. Вони мають зареєструватися в додатку та внести дані по якому предмету можуть допомогти.

Учні, та їх батьки зацікавлені у швидкому та зручному доступі до інформації. Цінність інформації у світі дуже висока. Роль розпорядників інформації у світі найчастіше виконують бази даних. Бази даних забезпечують надійне зберігання інформації, структурованому вигляді та своєчасний доступ до неї. Практично будь-яка сучасна організація потребує бази даних, що задовольняє ті чи інші потреби щодо зберігання, управління та адміністрування даних. У процесі виконання кваліфікаційної роботи виконана наступна робота:

- проведено аналіз та порівняння існуючих додатків прошуку викладачів;
- проаналізовані основні мови програмування;
- проаналізовані середовища розробки програмного забеспечення;
- розроблено базу даних;
- розроблено клієнтську на серверну частини програми.

В роботі проведено значний аналітичний огляд літературних джерел, досліджені основні підходи, що існують на сьогодні в областях розробки програмного забезпечення. На основі аналізу спроектовано і розроблено базу даних, серверну та клієнтську частини.

Користуючись даним додатком клієнти можуть швидко знайти потрібного викладача. Для цього потрібна загальна база даних, що включає всю потрібну інформацію. Програма дуже актуальна на сьогоднішній день, вона автоматизує роботу з базою даних і надає користувачеві (оператору) зрозумілий і дружній інтерфейс.

Завдяки дуже зручному об'єктно-орієнтованому дизайну, мова C# є хорошим вибором для швидкого створення високого рівня системних додатків, які використовують низькорівневий код. Також слід зазначити, що C# також веб-орієнтований – використовуючи прості вбудовані мовні конструкції, компоненти можна легко перетворити в веб-сервіси, доступ до яких можна отримати з Інтернету через будь-яку мову в будь-якій операційній системі.

В наслідок виконання проєкту було виконано систематизацію, закріплення, розширення теоретичних і практичних знань і вмінь зі спеціальності та застосування їх під час вирішення задач у самостійній практичній діяльності, розвинув навички самостійної роботи й оволодіння методиками, що пов’язані з виконанням виробничих функцій та типових задач діяльності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Типи додатків. URL: http://ipkey.com.ua/uk/faq/984-application.html (дата звернення 30.01.2022).
2. Wolfengagen V. E. Конструкції мов програмування. Методи опису. М.: Центр ЮрІнфо Р., 2001. 276 с.
3. Роберт В. Себеста. Основні поняття мов програмування . Поняття мов програмування. 5-е видання. М.: Вільямс, 2001. 672 р.
4. Д.Боуман, С.Эмерсон, М.Дарновський, “Практичний посібник по SQL”, “Діалектика”. Київ. 1997.
5. Oracle Java Technologies | Oracle. URL: http://java.sun.com/products/jdk/ (Дата звернення 05.02.2022)
6. Вікіпедія http://ru.wikipedia.org/wiki/Visual_Studio#Visual_Studio_2010 (Дата звернення 01.03.2022).
7. C# и .NET | Начало работы с Visual Studio. URL:https://metanit.com/sharp/tutorial/1.2.php (дата звернення 01.03.2022).
8. Троелсен Э. Мова програмування C# та платформа .NET 4.5 / Эндрю Троелсен. М: Вільямс, 2014.
9. Особливості тестування десктопних додатків у порівнянні з Web та мобільними додатками. URL: https://www.quality-assurance-group.com/osoblyvosti-testuvannya-desktopnyh-dodatkiv-u-porivnyanni (дата звернення 13.04.2022).

ДОДАТКИ

ДОДАТОК А
Лістинг класа Database

```
using System;
using System.Data.SqlClient;

namespace test_DataBase
{
    internal class DataBase
    {
        static String MyDataSource = "DESKTOP-VPLSMIC";
        static String MyBD = "Diplom_BD";
        SqlConnection sqlConnection = new SqlConnection(@"Data Source
=" + MyDataSource + ";Initial Catalog=" + MyBD + ";Integrated Secu-
rity=True");

        public void openConnection()
        {
            if (sqlConnection.State == System.Data.Connection-
State.Closed)
            {
            sqlConnection.Open();
            }
        }

        public void closeConnection()
        {
            if (sqlConnection.State == System.Data.Connection-
State.Open)
            {
                sqlConnection.Close();
            }
        }

        public SqlConnection getConnection()
        {
        return sqlConnection;
        }
    }
}
```

ДОДАТОК Б
Лістинг класа Form1

```
using MaterialSkin;
using MaterialSkin.Controls;
using System;
using System.Data;
using System.Data.SqlClient;
using System.IO;
using System.Windows.Forms;

namespace test_DataBase
{
    public partial class Form1 : MaterialForm
    {
        String MyTable = "Log_Table";
        String MyTable2 = "Teach_Card";
        String MyTable3 = "Student_Card";

        DataBase dataBase = new DataBase();
        SqlDataAdapter adapter = new SqlDataAdapter();
        DataTable dt = new DataTable();

        Form2 frm2 = new Form2();
        public int CurrentID = 0;
        int remember = 1;

        public Form1()
        {
            InitializeComponent();
            var materialSkinManager = MaterialSkinManager.Instance;
            materialSkinManager.AddFormToManage(this);
            materialSkinManager.Theme = MaterialSkinMan-
ager.Themes.LIGHT;
            materialSkinManager.ColorScheme = new ColorScheme(Pri-
mary.BlueGrey800, Primary.Red800, Primary.BlueGrey500, Ac-
cent.LightBlue200, TextShade.WHITE);
        }

        private void Form1_Load(object sender, EventArgs e) {}

        private void login_btn_Click(object sender, EventArgs e)
        {
            var login = login_text.Text;
            var pass = pass_text.Text;
            string sql_log = $"select ID_user, Login, Password from
{MyTable} where Login = '{login}' and Password = '{pass}'";

            SqlCommand cmd_log = new SqlCommand(sql_log, data-
Base.getConnection());
            adapter.SelectCommand = cmd_log;
            adapter.Fill(dt);

            if (dt.Rows.Count >= 1)
            {
                dataBase.openConnection();
                CurrentID = (int)cmd_log.ExecuteScalar();
                dataBase.closeConnection();
                MessageBox.Show("Ви успішно увійшли!", "Успішно!", Mes-
sageBoxButtons.OK, MessageBoxIcon.Information);

                frm2.curId = CurrentID;
                frm2.Show();
                dt.Rows.Clear();
```

```
                this.Hide();
            }
            else
                MessageBox.Show("Такого облікового запису не існує!",
"Облікового запису не існує!", MessageBoxButtons.OK, MessageBoxI-
con.Warning);
        }

        private void reg_btn_Click(object sender, EventArgs e)
        {
            if ((reg_pass_text1.Text == reg_pass_text2.Text) &&
(reg_pass_text1.Text != "") && (reg_login_text.Text != ""))
            {
                var login = reg_login_text.Text;
                var pass = reg_pass_text1.Text;
                int status = 0;
                int remem = 0;
                int getID = 0;
                String strBLOBFilePath = @"..\..\image\image2.png";

                //Зчитуємо jpg в файловий потік, а з файлового потоку в
байтовий масив
                FileStream fsBLOBFile = new FileStream(strBLOBFilePath,
FileMode.Open, FileAccess.Read);
                Byte[] bytBLOBData = new Byte[fsBLOBFile.Length];
                fsBLOBFile.Read(bytBLOBData, 0, bytBLOBData.Length);
                fsBLOBFile.Close();

                if (status_btn1.Checked == true)
                    status = 0;
                else
                    status = 1;

                string sql_verify = $"select Login from {MyTable} where
Login = '{login}'";
                string sql_cod = $"insert into {MyTable}(Login, Pass-
word, Status, Remeber) values ('{login}','{pass}','{sta-
tus}','{remem}')";
                string sql_getID = $"select ID_user from {MyTable} where
Login = '{login}'";
                string sql_regCard = "";

                SqlCommand cmd_verify = new SqlCommand(sql_verify, da-
taBase.getConnection());
                SqlCommand cmd_cod = new SqlCommand(sql_cod, data-
Base.getConnection());
                SqlCommand cmd_getID = new SqlCommand(sql_getID, data-
Base.getConnection());

                dt.Rows.Clear();
                adapter.SelectCommand = cmd_verify;
                adapter.Fill(dt);

                if (dt.Rows.Count == 0)
                {
                    dataBase.openConnection();

                    if (cmd_cod.ExecuteNonQuery() == 1)
                    {
                        getID = (int)cmd_getID.ExecuteScalar();

                        if (status == 1)
                        {
                            sql_regCard = $"insert into
{MyTable2}(ID_user, Name, SecondName, ThirdName, Description, Image,
Favor) values ('{getID}','','','','',@Img,'')";
```

```
                        }
                        else
                        {
                            sql_regCard      =      $"insert      into
{MyTable3}(ID_user, Name, SecondName, ThirdName, Status, Grade, Image,
Favor) values ('{getID}','','','','0','0',@Img,'')";
                        }

                        SqlCommand   cmd_regCard   =   new   SqlCom-
mand(sql_regCard, dataBase.getConnection());
                        SqlParameter  prm  =  new  SqlParameter("@Img",
SqlDbType.VarBinary,   bytBLOBData.Length,   ParameterDirection.Input,
false, 0, 0, null, DataRowVersion.Current, bytBLOBData);
                        cmd_regCard.Parameters.Add(prm);

                        if (cmd_regCard.ExecuteNonQuery() == 1)
                        {
                            MessageBox.Show("Успішна    регістрація!",
"Успішно!");
                            reg_login_text.Text = "";
                            reg_pass_text1.Text = "";
                            reg_pass_text2.Text = "";
                        }
                        else
                        {
                            MessageBox.Show("Реєстрація неуспішна!");
                        }
                    }
                    else
                    {
                        MessageBox.Show("Аккаунт не створено!");
                    }
                    dataBase.closeConnection();
                }
                else
                    MessageBox.Show("Такий логін вже існує", "Обліковий
запис вже існує!", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
            else
                MessageBox.Show("Аккаунт не створено!");
        }

    }
}
```

ДОДАТОК В
Лістинг класа Form2

```
using MaterialSkin;
using MaterialSkin.Controls;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Windows.Forms;
using System.Text.RegularExpressions;

namespace test_DataBase
{
    public partial class Form2 : MaterialForm
    {

        String MyTable = "Log_Table";
        String MyTable2 = "Teach_Card";
        String MyTable3 = "Student_Card";
        String MyTable4 = "Subj_Card";
        String MyTable5 = "Message_Table";

        int activeStr1 = 1,
            activeStr2 = 0,
            activeStr3 = 0;

        DataBase dataBase1 = new DataBase();
        SqlDataAdapter adapter1 = new SqlDataAdapter();
        DataTable dt = new DataTable();

        int status = -1;
        public int curId = 0;
        public int access = 0;
        public int accessStud = 0;
        int teachIdMsg = 0;
        int[] id_Access;
        int[] id_AccessStud;
        int currentAccess = 0;
        int myIdAccess = 0;
        int countMyMessage = 0;

        string firstName, secondName, thirdName, description;
        int studStatus, studGrade;

      static string[] subjNameArr = { "Математика", "Фізика", "Англійська", "Хімія", "Біологія", "Географія", "Укр мова", "Укр літ", "Інформатика" }; // сюди добавляємо нові предмети
      static string[] subjIndivArr = { "Групові", "Індивідуальні" };

      static string[] studStatusArr = { "Школяр (1-7клас)", "Старшокласник (8-11клас)", "Студент" };
      static string[] studGrade7Arr = { "1", "2", "3", "4", "5", "6", "7" }; // класи з 1 по 7
      static string[] studGrade11Arr = { "8", "9", "10", "11" }; // класи з 8 по 11
      static string[] studGradeKursArr = { "1", "2", "3", "4", "5", "6" }; // курси з 1 по 6
```

```
        int[] countSubjGlob = new int[subjNameArr.Length];
        static int countObj = subjNameArr.Length;

        CheckBox[] checkObj = new CheckBox[countObj];
        CheckBox[] checkIndiv = new CheckBox[2];
        Panel panelMsgMain = new Panel();
        RichTextBox textMsgBox = new RichTextBox();

        public Form2()
        {
            InitializeComponent();
            var materialSkinManager = MaterialSkinManager.Instance;
            materialSkinManager.AddFormToManage(this);
            materialSkinManager.Theme                                    =
MaterialSkinManager.Themes.LIGHT;
            materialSkinManager.ColorScheme          =          new
ColorScheme(Primary.BlueGrey800,                  Primary.Red600,
Primary.BlueGrey500, Accent.Red700, TextShade.WHITE);

            for (int i = 0; i < subjNameArr.Length; i++)
            {
                Subj_Column.Items.Add(subjNameArr[i]);
            }

            for (int i = 0; i < subjIndivArr.Length; i++)
            {
                TypeTeach_Column.Items.Add(subjIndivArr[i]);
            }

            for (int i = 0; i < studStatusArr.Length; i++)
            {
                studStatus_Box.Items.Add(studStatusArr[i]);
            }

            tabPage1.Controls.Add(panelStr1);
            tabPage1.Controls.Add(panelStr1_1);
            tabPage1.Controls.Add(panelStr1_2);
            panelStr1_2_3.BackColor = Color.FromArgb(55, 71, 79);

            checkObj[0] = checkBox_Math;
            checkObj[1] = checkBox_Xim;
            checkObj[2] = checkBox_Phyz;
            checkObj[3] = checkBox_Biol;
            checkObj[4] = checkBox_Geogr;
            checkObj[5] = checkBox_UkrMov;
            checkObj[6] = checkBox_UkrLit;
            checkObj[7] = checkBox_Inform;
            checkObj[8] = checkBox_Eng;

            checkIndiv[0] = checkBox_Group;
            checkIndiv[1] = checkBox_Indiv;

            if (checkBox_All.Checked == true)
            {
                for (int i = 0; i < countObj; i++) {
                    checkObj[i].Checked = true;
                }
                for (int i = 0; i < 2; i++)
                {
                    checkIndiv[i].Checked = true;
                }
            }
        }

        private      void      Form2_FormClosing(object      sender,
FormClosingEventArgs e)
```

```
        {
            Application.Exit();
        }

        private void Form2_Load(object sender, EventArgs e)
        {

this.subj_CardTableAdapter.Fill(this.diplom_BDDataSet.Subj_Card);
            dataBase1.openConnection();

            // команди для Бази даних
            string sql_Status = $"SELECT Status FROM {MyTable} WHERE
ID_user = {curId}";
            string sql_Teach = $"SELECT Image FROM {MyTable2} WHERE
ID_user = {curId}";
            string  sql_NameTeach  =  $"SELECT  Name,  SecondName,
ThirdName, Description FROM {MyTable2} WHERE ID_user = {curId}";
            string sql_Stud = $"SELECT Image FROM {MyTable3} WHERE
ID_user = {curId}";
            string  sql_NameStud  =  $"SELECT  Name,  SecondName,
ThirdName, Status, Grade FROM {MyTable3} WHERE ID_user = {curId}";
            string sql_SubjTable = $"SELECT Subj_Name, TypeTeach,
Cost, Staj, Deskription FROM {MyTable4} WHERE ID_user = {curId}";
            string  sql_CountSubj  =  $"SELECT  COUNT(ID_Subj)  FROM
{MyTable4} WHERE ID_user = {curId}";
            string sql_AccessTeach = $"SELECT ID_user FROM {MyTable}
WHERE Remeber = 1";
            string sql_CountAccess = $"SELECT COUNT(ID_user) FROM
{MyTable} WHERE Remeber = 1";
            string sql_AccessStud = $"SELECT ID_user FROM {MyTable}
WHERE Remeber = 2";
            string  sql_CountAccessStud  =  $"SELECT  COUNT(ID_user)
FROM {MyTable} WHERE Remeber = 2";

            SqlCommand  cmd_Status  =  new  SqlCommand(sql_Status,
dataBase1.getConnection());
            status = (int)cmd_Status.ExecuteScalar();

            SqlCommand command1, command2, command3;
            string sortTable;

            if (status == 1) {
                sortTable = MyTable2;
                command1      =      new      SqlCommand(sql_Teach,
dataBase1.getConnection());
                command2     =     new     SqlCommand(sql_NameTeach,
dataBase1.getConnection());

                materialLabel6.Visible = true;
                DescriptBox.Visible = true;
                dell_Btn.Visible = true;
                dataGridView1.Visible = true;

                studStatus_Box.Visible = false;
                studGrade_Box.Visible = false;
                studStatus_Lable.Visible = false;
                studGrade_Lable.Visible = false;

            }
            else {
                sortTable = MyTable3;
                command1      =      new      SqlCommand(sql_Stud,
dataBase1.getConnection());
                command2     =     new     SqlCommand(sql_NameStud,
dataBase1.getConnection());
                materialLabel6.Visible = false;
```

```
            DescriptBox.Visible = false;
            dell_Btn.Visible = false;
            dataGridView1.Visible = false;

            studStatus_Box.Visible = true;
            studGrade_Box.Visible = true;
            studStatus_Lable.Visible = true;
            studGrade_Lable.Visible = true;
        }

        SqlDataAdapter da = new SqlDataAdapter(command1);
        DataSet ds = new DataSet();
        da.Fill(ds, sortTable);
        int c = ds.Tables[sortTable].Rows.Count;

        if (c > 0)
        {
            Byte[] byteBLOBData = new Byte[0];
            byteBLOBData = (Byte[])(ds.Tables[sortTable].Rows[c
- 1]["Image"]);
            MemoryStream        stmBLOBData        =        new
MemoryStream(byteBLOBData);
            pictureBox2.Image = Image.FromStream(stmBLOBData);
        }

        dataGridView1.Rows.Clear();

        SqlDataAdapter adapter2 = new SqlDataAdapter();
        DataSet dataSet2 = new DataSet();
        adapter2.SelectCommand = command2;
        adapter2.Fill(dataSet2);
        SqlDataReader reader2 = command2.ExecuteReader();

        while (reader2.Read())
        {
            if (status == 1) {
                firstName = reader2.GetString(0);
                secondName = reader2.GetString(1);
                thirdName = reader2.GetString(2);
                description = reader2.GetString(3);
            }
            else {
                firstName = reader2.GetString(0);
                secondName = reader2.GetString(1);
                thirdName = reader2.GetString(2);
                studStatus = reader2.GetInt32(3);
                studGrade = reader2.GetInt32(4);
            }
        }
        reader2.Close();

        int countSubj;
        SqlCommand cmd_CountSubj = new SqlCommand(sql_CountSubj,
dataBase1.getConnection());
        countSubj = (int)cmd_CountSubj.ExecuteScalar();

        string[] subjName = new string[countSubj];
        string[] staj = new string[countSubj];
        string[] deskripSubj = new string[countSubj];
        int[] typeTeach = new int[countSubj];
        int[] cost = new int[countSubj];

        if (status == 1)
        {
            SqlCommand        cmd_SubjTable        =        new
SqlCommand(sql_SubjTable, dataBase1.getConnection());
```

```
                SqlDataAdapter adapter3 = new SqlDataAdapter();
                DataSet dataSet3 = new DataSet();
                adapter3.SelectCommand = cmd_SubjTable;
                adapter3.Fill(dataSet3);
                SqlDataReader               reader3               =
cmd_SubjTable.ExecuteReader();

                int ki = 0;
                while (reader3.Read())
                {
                    subjName[ki] = reader3.GetString(0);
                    typeTeach[ki] = reader3.GetInt32(1);
                    cost[ki] = reader3.GetInt32(2);
                    staj[ki] = reader3.GetString(3);
                    deskripSubj[ki] = reader3.GetString(4);
                    ki++;
                }
                reader3.Close();

                for (int i = 0; i < subjNameArr.Length; i++)
                {
                    string sql_countSubj = $"SELECT COUNT(ID_subj)
FROM  {MyTable4}  WHERE  ID_user  =  {curId}  and  Subj_Name  =
'{subjNameArr[i]}'";
                    SqlCommand      cmd_countSubj      =      new
SqlCommand(sql_countSubj, dataBase1.getConnection());
                    countSubjGlob[i]                                =
(int)cmd_countSubj.ExecuteScalar();
                }
            }

            dataBase1.closeConnection();
            studGrade_Box.Items.Clear();

            if (status == 1)
            {
                NameText.Text = firstName;
                SecondNameText.Text = secondName;
                ThirdNameText.Text = thirdName;
                DescriptBox.Text = description;
            }
            else
            {
                NameText.Text = firstName;
                SecondNameText.Text = secondName;
                ThirdNameText.Text = thirdName;
                if (studStatus >= 1)
                    studStatus_Box.SelectedIndex = studStatus - 1;

                if (studStatus_Box.SelectedIndex == 0)
                {
                    for (int i = 0; i < studGrade7Arr.Length; i++)
                    {
                        studGrade_Box.Items.Add(studGrade7Arr[i]);
                    }
                    if (studGrade >= 1)
                    {
                        studGrade_Box.SelectedIndex = studGrade - 1;
                    }
                }
                else if (studStatus_Box.SelectedIndex == 1)
                {
                    for (int i = 0; i < studGrade11Arr.Length; i++)
                    {
                        studGrade_Box.Items.Add(studGrade11Arr[i]);
                    }
```

```
                    if (studGrade >= 1)
                    {
                        studGrade_Box.SelectedIndex = studGrade - 1;
                    }
                }
                else if (studStatus_Box.SelectedIndex == 2)
                {
                    for (int i = 0; i < studGradeKursArr.Length; i++)
                    {
studGrade_Box.Items.Add(studGradeKursArr[i]);
                    }
                    if (studGrade >= 1)
                    {
                        studGrade_Box.SelectedIndex = studGrade - 1;
                    }
                }
            }

            int rows = dataGridView1.Rows.Count;

            if (status == 1)
            {
                if (rows <= countSubj)
                {
                    for (int i = 0; i < countSubj; i++)
                    {
                        dataGridView1.Rows.Add();
                    }
                }

                for (int i = 0; i < countSubj; i++)
                {
                    dataGridView1.Rows[i].Cells[0].Value          =
subjName[i];
                    if (typeTeach[i] == 1)
                    {
                        dataGridView1.Rows[i].Cells[1].Value      =
subjIndivArr[0];    // групові
                    }
                    else if (typeTeach[i] == 2)
                    {
                        dataGridView1.Rows[i].Cells[1].Value      =
subjIndivArr[1];    // індивідуальні
                    }

                    dataGridView1.Rows[i].Cells[2].Value = cost[i];
                    dataGridView1.Rows[i].Cells[3].Value = staj[i];
                    dataGridView1.Rows[i].Cells[4].Value          =
deskripSubj[i];
                }
            }

            dataBase1.openConnection();

            SqlCommand cmd_UpdAccess;
            if (status == 1)
            {
                if (firstName != "" && secondName != "" && thirdName
!= "" && description != "" && countSubj >= 1)
                {
                    string sql_UpdAccess = $"UPDATE {MyTable} SET
Remeber = 1 WHERE ID_user = {curId};";
                    cmd_UpdAccess = new SqlCommand(sql_UpdAccess,
dataBase1.getConnection());
                    cmd_UpdAccess.ExecuteNonQuery();
```

```
            }
            else if (firstName == "" || secondName == "" ||
thirdName == "" || description == "" || countSubj < 1)
            {
                string sql_UpdAccess = $"UPDATE {MyTable} SET
Remeber = 0 WHERE ID_user = {curId};";
                cmd_UpdAccess = new SqlCommand(sql_UpdAccess,
dataBase1.getConnection());
                cmd_UpdAccess.ExecuteNonQuery();
            }
        }
        else if (status == 0)
        {
            if (firstName != "" && secondName != "" && thirdName
!=    ""    &&    studStatus_Box.SelectedIndex    >=    0    &&
studGrade_Box.SelectedIndex >= 0)
            {
                string sql_UpdAccess = $"UPDATE {MyTable} SET
Remeber = 2 WHERE ID_user = {curId};";
                cmd_UpdAccess = new SqlCommand(sql_UpdAccess,
dataBase1.getConnection());
                cmd_UpdAccess.ExecuteNonQuery();
            }
            else if (firstName == "" || secondName == "" ||
thirdName  ==  ""  ||  studStatus_Box.SelectedIndex  <  0  ||
studGrade_Box.SelectedIndex < 0)
            {
                string sql_UpdAccess = $"UPDATE {MyTable} SET
Remeber = 0 WHERE ID_user = {curId};";
                cmd_UpdAccess = new SqlCommand(sql_UpdAccess,
dataBase1.getConnection());
                cmd_UpdAccess.ExecuteNonQuery();
            }
        }

        SqlCommand        cmd_CountAccess        =        new
SqlCommand(sql_CountAccess, dataBase1.getConnection());
        access = (int)cmd_CountAccess.ExecuteScalar();
        id_Access = new int[access];

        command3     =     new     SqlCommand(sql_AccessTeach,
dataBase1.getConnection());
        SqlDataAdapter adapter4 = new SqlDataAdapter();
        DataSet dataSet4 = new DataSet();
        adapter4.SelectCommand = command3;
        adapter4.Fill(dataSet4);
        SqlDataReader reader4 = command3.ExecuteReader();

        int y = 0;
        while (reader4.Read())
        {
            id_Access[y] = reader4.GetInt32(0);
            y++;
        }
        reader4.Close();

        SqlCommand command5;
        SqlCommand       cmd_CountAccessStud       =       new
SqlCommand(sql_CountAccessStud, dataBase1.getConnection());
        accessStud = (int)cmd_CountAccessStud.ExecuteScalar();
        id_AccessStud = new int[accessStud];

        command5     =     new     SqlCommand(sql_AccessStud,
dataBase1.getConnection());
        SqlDataAdapter adapter5 = new SqlDataAdapter();
        DataSet dataSet5 = new DataSet();
```

```
            adapter5.SelectCommand = command5;
            adapter5.Fill(dataSet5);
            SqlDataReader reader5 = command5.ExecuteReader();

            int yy = 0;
            while (reader5.Read())
            {
                id_AccessStud[yy] = reader5.GetInt32(0);
                yy++;
            }
            reader5.Close();
            dataBase1.closeConnection();

            int currAccess = 0;

            if (status == 1)
            {
                for (int i = 0; i < access; i++)
                {
                    if (id_Access[i] == curId)
                        currAccess++;
                }

                if (currAccess == 0)
                    statusAccess_Lable.Text = "Заповніть усі поля
для опублікування Вашої картки!";
                else if (currAccess == 1)
                    statusAccess_Lable.Text = "";
            }
            else if (status == 0)
            {
                for (int i = 0; i < accessStud; i++)
                {
                    if (id_AccessStud[i] == curId)
                        currAccess++;
                }

                if (currAccess == 0)
                    statusAccess_Lable.Text = "Заповніть усі поля
для можливості написати викладачам!";
                else if (currAccess == 1)
                    statusAccess_Lable.Text = "";
            }

            myIdAccess = currAccess;

            int countChoiseSubj = 0;
            for (int i = 0; i < countObj; i++) {
                if (checkObj[i].Checked == true)
                    countChoiseSubj++;
            }

            int[] choiseSubjarr = new int[countChoiseSubj];
            string[] choiseSubjStr = new string[countChoiseSubj];
            int kj = 0;
            for (int i = 0; i < countObj; i++)
            {
                if (checkObj[i].Checked == true)
                {
                    choiseSubjarr[kj] = i;
                    choiseSubjStr[kj] = checkObj[i].Text;
                    kj++;
                }
            }

            int countIndiv = 0;
```

```
            for (int i = 0; i < 2; i++)
            {
                if (checkIndiv[i].Checked == true)
                    countIndiv++;
            }

            int[] choiseIndivArr = new int[countIndiv];
            int[] choiseIndivValue = new int[countIndiv];
            int kk = 0;
            int checkedIndiv = 0;
            for (int i = 0; i < 2; i++)
            {
                if (checkIndiv[i].Checked == true)
                {
                    choiseIndivArr[kk] = i;
                    if (checkIndiv[i].Text == subjIndivArr[0])    //
групові
                    {
                        choiseIndivValue[kk] = 1;
                    }
                    else if (checkIndiv[i].Text == subjIndivArr[1])
// індивідуальні
                    {
                        choiseIndivValue[kk] = 2;
                    }
                    kk++;
                }
                else
                {
                    checkedIndiv++;
                }
            }

            dataBase1.openConnection();

            int countSubjId = 0;
            string  sql_GetCountID  =  $"SELECT  COUNT(ID_subj)  FROM
{MyTable4}";
            SqlCommand          cmd_GetCountID          =          new
SqlCommand(sql_GetCountID, dataBase1.getConnection());
            countSubjId = (int)cmd_GetCountID.ExecuteScalar();

            if (maxCost_Field.Text == "") {
                maxCost_Field.Text = "100000";
            }

            int maxCost = Convert.ToInt32(maxCost_Field.Text);
            maxCost_Field.Clear();

            int[] getId = new int[countSubjId];
            int ka = 0;

            for (int i = 0; i < countChoiseSubj; i++)
            {
                if (checkedIndiv < 2)
                {
                    for (int j = 0; j < countIndiv; j++)
                    {
                        string  sql_GetID  =  $"SELECT  ID_user  FROM
{MyTable4} WHERE Subj_Name = '{choiseSubjStr[i]}' and TypeTeach =
{choiseIndivValue[j]} and Cost <= {maxCost}";
                        SqlCommand       cmd_GetID       =       new
SqlCommand(sql_GetID, dataBase1.getConnection());
                        SqlDataAdapter      adapter7      =      new
SqlDataAdapter();
                        DataSet dataSet7 = new DataSet();
```

```
                    adapter7.SelectCommand = cmd_GetID;
                    adapter7.Fill(dataSet7);
                    SqlDataReader reader7 =
cmd_GetID.ExecuteReader();

                    while (reader7.Read())
                    {
                        for (int yu = 0; yu < access; yu++) {
                            if (id_Access[yu] ==
reader7.GetInt32(0))
                            {
                                getId[ka] =
reader7.GetInt32(0);
                                ka++;
                            }
                        }
                    }
                    reader7.Close();
                }
            }
            else
            {
                string sql_GetID = $"SELECT ID_user FROM
{MyTable4} WHERE Subj_Name = '{choiseSubjStr[i]}' and Cost <=
{maxCost}";
                SqlCommand cmd_GetID = new SqlCommand(sql_GetID,
dataBase1.getConnection());
                SqlDataAdapter adapter7 = new SqlDataAdapter();
                DataSet dataSet7 = new DataSet();
                adapter7.SelectCommand = cmd_GetID;
                adapter7.Fill(dataSet7);
                SqlDataReader reader7 =
cmd_GetID.ExecuteReader();

                while (reader7.Read())
                {
                    for (int yu = 0; yu < access; yu++)
                    {
                        if (id_Access[yu] ==
reader7.GetInt32(0))
                        {
                            getId[ka] = reader7.GetInt32(0);
                            ka++;
                        }
                    }
                }
                reader7.Close();
            }
        }

        dataBase1.closeConnection();

        var ls = new List<int>();
        for (int i = 0; i < ka; i++)
        {
            ls.Add(getId[i]);
        }
        var sortId = ls.Distinct().ToArray();
        int[] sortedId = sortId;

        CreateMyPanel(sortedId);
        CreateMessagePanel();
    }


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

```
      {
          var name = NameText.Text;
          var secondName = SecondNameText.Text;
          var thirdName = ThirdNameText.Text;
          var descript = DescriptBox.Text;
          int studStatusUpd = studStatus_Box.SelectedIndex + 1;
          int studGradeUpd = studGrade_Box.SelectedIndex + 1;

          string sql_UpdateStudProfile = $"UPDATE {MyTable3} SET
Name  =  '{name}',  SecondName  =  '{secondName}',  ThirdName  =
'{thirdName}', Status = {studStatusUpd}, Grade = {studGradeUpd}
WHERE ID_user = {curId};";
          string sql_UpdateTeachProfile = $"UPDATE {MyTable2} SET
Name  =  '{name}',  SecondName  =  '{secondName}',  ThirdName  =
'{thirdName}',  Description  =  '{descript}'  WHERE  ID_user  =
{curId}";

          SqlCommand command;
          dataBase1.openConnection();

          if (status == 1)
          {
              command  =  new  SqlCommand(sql_UpdateTeachProfile,
dataBase1.getConnection());
              command.ExecuteNonQuery();
              MessageBox.Show("Збережено!",           "Успішно!",
MessageBoxButtons.OK, MessageBoxIcon.Information);
          }
          else
          {
              command  =  new  SqlCommand(sql_UpdateStudProfile,
dataBase1.getConnection());
              command.ExecuteNonQuery();
              MessageBox.Show("Збережено!",           "Успішно!",
MessageBoxButtons.OK, MessageBoxIcon.Information);
          }
          dataBase1.closeConnection();

          int countChek = 0;
          int countRepeat = 0;
          int rows = dataGridView1.Rows.Count;

          if (status == 1)
          {
              int[] check = new int[rows];
              dataGridView1.ClearSelection();

              for (int i = 0; i < rows - 1; i++)
              {
                  string                  subjName                =
Convert.ToString(dataGridView1.Rows[i].Cells[0].FormattedValue.To
String());
                  string              typeTeach_str               =
Convert.ToString(dataGridView1.Rows[i].Cells[1].FormattedValue.To
String());
                  string                    cost                  =
Convert.ToString(dataGridView1.Rows[i].Cells[2].FormattedValue.To
String());
                  string                    staj                  =
Convert.ToString(dataGridView1.Rows[i].Cells[3].FormattedValue.To
String());
                  string              descript_Subj               =
Convert.ToString(dataGridView1.Rows[i].Cells[4].FormattedValue.To
String());
                  if (subjName == "" || typeTeach_str == "" || cost
== "" || staj == "")
```

```
                    {
                        check[countChek] = i;
                        countChek++;
                        dataGridView1.Rows[i].Selected = true;
                    }
                    else
                    {
                        dataBase1.openConnection();

                        int typeTeach = 0;
                        if (typeTeach_str == "Групові")
                        {
                            typeTeach = 1;
                        }
                        else if (typeTeach_str == "Індивідуальні")
                        {
                            typeTeach = 2;
                        }

                        string sql_maxId = $"SELECT COUNT(ID_subj)
FROM {MyTable4} WHERE ID_user = {curId}";
                        SqlCommand       cmd_maxId       =       new
SqlCommand(sql_maxId, dataBase1.getConnection());
                        int maxId = (int)cmd_maxId.ExecuteScalar();

                        int[]         countSubj         =         new
int[subjNameArr.Length];
                        for (int j = 0; j < subjNameArr.Length; j++)
                        {
                            string    sql_countSubj    =    $"SELECT
COUNT(ID_subj)  FROM  {MyTable4}  WHERE  ID_user  =  {curId}  and
Subj_Name = '{subjNameArr[j]}'";
                            SqlCommand    cmd_countSubj    =    new
SqlCommand(sql_countSubj, dataBase1.getConnection());
                            countSubj[j]                            =
(int)cmd_countSubj.ExecuteScalar();
                        }

                        if (rows - 1 > maxId && i >= maxId)
                        {
                            int newSubj = maxId + 1;
                            for (int k = 0; k < subjNameArr.Length;
k++)
                            {
                                if  (subjName  ==  subjNameArr[k]  &&
countSubj[k] >= 2)
                                {
                                    dataGridView1.Rows[i].Selected
= true;
                                }
                                else if (subjName == subjNameArr[k]
&& countSubj[k] < 2)
                                {
                                    string  sql_addRow  =  $"insert
into  {MyTable4}  (ID_user,  ID_subj,  Subj_Name,  TypeTeach,  Cost,
Staj,                   Deskription)                    values
('{curId}','{newSubj}','{subjName}','{typeTeach}','{Convert.ToInt
16(cost)}','{staj}','{descript_Subj}')";
                                    SqlCommand  cmd_addRow  =  new
SqlCommand(sql_addRow, dataBase1.getConnection());
                                    cmd_addRow.ExecuteNonQuery();
                                }
                            }
                        }
                        else if (rows - 1 == maxId)
                        {
```

```
                            for (int k = 0; k < subjNameArr.Length;
k++)
                            {
                                if (subjName == subjNameArr[k] &&
countSubj[k] == countSubjGlob[k] && countSubj[k] >= 2)
                                {
                                    string sql_UpdateSubj = $"UPDATE
{MyTable4}    SET    TypeTeach    =    '{typeTeach}',    Cost    =
'{Convert.ToInt16(cost)}',   Staj   =   '{staj}',   Deskription   =
'{descript_Subj}' WHERE ID_user = {curId} and ID_subj = {i + 1}";
                                    SqlCommand cmd_UpdateSubj = new
SqlCommand(sql_UpdateSubj, dataBase1.getConnection());

cmd_UpdateSubj.ExecuteNonQuery();
                                }
                                else if (subjName == subjNameArr[k]
&& countSubj[k] == countSubjGlob[k] && countSubj[k] <= 2)
                                {
                                    string sql_UpdateSubj = $"UPDATE
{MyTable4}   SET   Subj_Name   =   '{subjName}',   TypeTeach   =
'{typeTeach}', Cost = '{Convert.ToInt16(cost)}', Staj = '{staj}',
Deskription = '{descript_Subj}' WHERE ID_user = {curId} and ID_subj
= {i + 1}";
                                    SqlCommand cmd_UpdateSubj = new
SqlCommand(sql_UpdateSubj, dataBase1.getConnection());

cmd_UpdateSubj.ExecuteNonQuery();
                                }
                                else
                                    countRepeat++;
                            }
                        }
                        dataBase1.closeConnection();
                    }
                }
            }

            if (countChek > 0)
            {
                MessageBox.Show("Ви заповнили не всі дані, щоб до-
дати         предмет!",          "",          MessageBoxButtons.OK,
MessageBoxIcon.Information);
            }
            else if (countRepeat > 0)
            {
                Form2_Load(sender, e);
            }
            else
            {
                Form2_Load(sender, e);
            }
        }


        private void dell_Btn_Click(object sender, EventArgs e)
        {
            int check = 0;
            int rows = dataGridView1.Rows.Count;
            int[] delete = new int[rows];
            int deleteCount = 0;

            dataBase1.openConnection();
            string   sql_maxId   =   $"SELECT   COUNT(ID_subj)   FROM
{MyTable4} WHERE ID_user = {curId}";
            SqlCommand   cmd_maxId   =   new   SqlCommand(sql_maxId,
dataBase1.getConnection());
```

```
            int maxId = (int)cmd_maxId.ExecuteScalar();
            if (rows - 1 == maxId)
                check = 1;

            for (int i = 0; i < rows - 1; i++)
            {
                if (dataGridView1.Rows[i].Selected == true)
                {
                    string sql_DeleteSubj = $"DELETE FROM {MyTable4}
WHERE ID_user = {curId} and ID_subj = {i + 1}";
                    SqlCommand      cmd_DeleteSubj      =      new
SqlCommand(sql_DeleteSubj, dataBase1.getConnection());
                    cmd_DeleteSubj.ExecuteNonQuery();
                    delete[i] = i + 1;
                    deleteCount++;
                }
            }

            string  sql_maxId1  =  $"SELECT  COUNT(ID_subj)  FROM
{MyTable4} WHERE ID_user = {curId}";
            SqlCommand  cmd_maxId1  =  new  SqlCommand(sql_maxId1,
dataBase1.getConnection());
            int maxId1 = (int)cmd_maxId1.ExecuteScalar();
            int newCount = 0;

            for (int i = 0; i < maxId; i++)
            {
                if (maxId1 != maxId)
                {
                    if (delete[i] > 0)
                        newCount++;
                    else if (delete[i] == 0)
                    {
                        string sql_UpdateSubj = $"UPDATE {MyTable4}
SET ID_subj = {i + 1 - newCount} WHERE ID_user = {curId} and ID_subj
= {i + 1}";
                        SqlCommand     cmd_UpdateSubj     =     new
SqlCommand(sql_UpdateSubj, dataBase1.getConnection());
                        cmd_UpdateSubj.ExecuteNonQuery();
                    }
                }
            }
            dataBase1.closeConnection();

            Form2_Load(sender, e);
        }

        public void CreateMyPanel(int[] sortedId)
        {
            panelStr1.BackColor = Color.FromArgb(55, 71, 79);

            int countSubjName = 0;
            string[] FIO = new string[sortedId.Length];
            string[] subjName = new string[sortedId.Length];
            int[] maxCost = new int[sortedId.Length],
                  minCost = new int[sortedId.Length],
                  avgCost = new int[sortedId.Length];
            string[] otzivStr = new string[sortedId.Length];
            Byte[][] byteImg = new Byte[sortedId.Length][];

            Panel[] panel3 = new Panel[sortedId.Length];
            PictureBox[]          pictureBox1          =          new
PictureBox[sortedId.Length];
            Label[] label1 = new Label[sortedId.Length];
            Label[] label2 = new Label[sortedId.Length];
            Label[] label3 = new Label[sortedId.Length];
```

```
            Label[] label4 = new Label[sortedId.Length];

            dataBase1.openConnection();
            for (int i = 0; i < sortedId.Length; i++)
            {
                SqlCommand cmd_NameTeach = new SqlCommand($"SELECT
Name, SecondName, ThirdName FROM {MyTable2} WHERE ID_user =
{sortedId[i]}", dataBase1.getConnection());
                SqlDataAdapter adapter3 = new SqlDataAdapter();
                DataSet dataSet3 = new DataSet();
                adapter3.SelectCommand = cmd_NameTeach;
                adapter3.Fill(dataSet3);
                SqlDataReader               reader3               =
cmd_NameTeach.ExecuteReader();

                while (reader3.Read())
                {
                    FIO[i]  =  reader3.GetString(0)  +  "  "  +
reader3.GetString(1) + " " + reader3.GetString(2);
                }
                reader3.Close();

                SqlCommand cmd_SubjTable = new SqlCommand($"SELECT
COUNT(DISTINCT  Subj_Name)  FROM  {MyTable4}  WHERE  ID_user  =
{sortedId[i]}", dataBase1.getConnection());
                countSubjName = (int)cmd_SubjTable.ExecuteScalar();

                SqlCommand cmd_SubjTable1 = new SqlCommand($"SELECT
DISTINCT Subj_Name FROM {MyTable4} WHERE ID_user = {sortedId[i]}",
dataBase1.getConnection());
                SqlDataAdapter adapter6 = new SqlDataAdapter();
                DataSet dataSet6 = new DataSet();
                adapter6.SelectCommand = cmd_SubjTable1;
                adapter6.Fill(dataSet6);
                SqlDataReader               reader6               =
cmd_SubjTable1.ExecuteReader();

                int j = 0;
                int repet = 0;
                while (reader6.Read())
                {
                    if (j >= 10 && repet >= 10) {
                        subjName[i] += "\n" + reader6.GetString(0) +
", ";
                        repet = 0;
                    }
                    else if (j == countSubjName - 1) {
                        subjName[i] += reader6.GetString(0);
                    }
                    else
                        subjName[i] += reader6.GetString(0) + ", ";
                    repet++;
                    j++;
                }
                reader6.Close();

                SqlCommand  cmd_MaxCost  =  new  SqlCommand($"SELECT
MAX(Cost)  FROM  {MyTable4}  WHERE  ID_user  =  {sortedId[i]}",
dataBase1.getConnection());
                SqlCommand  cmd_AvgCost  =  new  SqlCommand($"SELECT
AVG(Cost)  FROM  {MyTable4}  WHERE  ID_user  =  {sortedId[i]}",
dataBase1.getConnection());
                SqlCommand  cmd_MinCost  =  new  SqlCommand($"SELECT
MIN(Cost)  FROM  {MyTable4}  WHERE  ID_user  =  {sortedId[i]}",
dataBase1.getConnection());
```

```
            maxCost[i] = (int)cmd_MaxCost.ExecuteScalar();
            avgCost[i] = (int)cmd_AvgCost.ExecuteScalar();
            minCost[i] = (int)cmd_MinCost.ExecuteScalar();

            SqlCommand cmd_GetOtziv = new SqlCommand($"SELECT
Favor FROM {MyTable2} WHERE ID_user = {sortedId[i]}",
dataBase1.getConnection());
            string otzivi =
(string)cmd_GetOtziv.ExecuteScalar();

            String requiredString = otzivi;
            int countOtziv = new
Regex("id:").Matches(otzivi).Count;

            int summ = 0;
            while (requiredString.IndexOf(",") >= 0)
            {
                int firstPos = requiredString.IndexOf(",") + 1;
                int secondPos = requiredString.IndexOf(";");

                summ +=
Convert.ToInt32(requiredString.Substring(firstPos, secondPos -
firstPos));
                requiredString =
requiredString.Substring(requiredString.IndexOf(";") + 1);
            }

            float avgOtziv = 0;
            if (countOtziv > 0)
            {
                avgOtziv = (float)summ / (float)countOtziv;
                otzivStr[i] = "Відгуки(" + countOtziv + "): " +
avgOtziv.ToString("#.#") + " з 5";
            }
            else
            {
                otzivStr[i] = "Відгуки(" + countOtziv + ")";
            }

            SqlCommand cmd_getImg = new SqlCommand($"SELECT
Image FROM {MyTable2} WHERE ID_user = {sortedId[i]}",
dataBase1.getConnection());
            SqlDataAdapter da = new SqlDataAdapter(cmd_getImg);
            DataSet ds = new DataSet();
            da.Fill(ds, MyTable2);
            int c = ds.Tables[MyTable2].Rows.Count;

            if (c > 0)
            {
                byteImg[i] =
(Byte[])(ds.Tables[MyTable2].Rows[c - 1]["Image"]);
            }
        }

        dataBase1.closeConnection();

        int countY = 0;
        int countOtstupY = 0;
        int countOtstupX = 0;

        for (int i = 0; i < sortedId.Length; i++)
        {
            int panel_X = 100,
                panel_Y = 350,
                panel_OtstupX = 47,
                panel_OtstupY = 40,
```

```
                panel_SizeX = 270,
                panel_SizeY = 335;

            if (countY >= 3)
            {
                countY = 0;
                countOtstupY++;
                countOtstupX = 0;
            }

            panel3[i] = new Panel();
            panel3[i].Location  =  new  Point(panel_OtstupX  *
(countOtstupX + 1) + (panel_SizeX * countOtstupX), panel_OtstupY
* (countOtstupY + 1) + (panel_SizeY * countOtstupY));
            panel3[i].Size = new Size(panel_SizeX, panel_SizeY);
            panel3[i].BorderStyle                                =
System.Windows.Forms.BorderStyle.Fixed3D;
            panel3[i].BackColor = Color.White;

            countY++;
            countOtstupX++;

            pictureBox1[i] = new PictureBox();
            MemoryStream        stmBLOBData        =        new
MemoryStream(byteImg[i]);
            pictureBox1[i].Image                                 =
Image.FromStream(stmBLOBData);
            pictureBox1[i].Size = new Size(170, 195);
            pictureBox1[i].AutoSize = true;
            pictureBox1[i].SizeMode = PictureBoxSizeMode.Zoom;
            pictureBox1[i].Left     =     (panel3[i].Width     -
pictureBox1[i].Width) / 2;
            pictureBox1[i].Top = 10;

            label1[i] = new Label();
            label1[i].Text = FIO[i];
            label1[i].TextAlign                                  =
ContentAlignment.MiddleCenter;
            label1[i].Font    =    new    Font("Arial",    11,
FontStyle.Regular);
            label1[i].MaximumSize  =  new  Size(panel3[i].Width,
label1[i].Height);
            label1[i].Size                                       =
TextRenderer.MeasureText(label1[i].Text + ".", label1[i].Font);
            label1[i].Left = (panel3[i].Width - label1[i].Width)
/ 2;
            label1[i].Top = pictureBox1[i].Bottom + 5;

            label2[i] = new Label();
            label2[i].Text = "Предмети: " + subjName[i];
            label2[i].TextAlign                                  =
ContentAlignment.MiddleCenter;
            label2[i].Font    =    new    Font("Arial",    11,
FontStyle.Regular);
            label2[i].Size                                       =
TextRenderer.MeasureText(label2[i].Text + ".", label2[i].Font);
            int   height1   =   (int)((int)(label2[i].Width   /
panel3[i].Width) + 1);
            if (height1 > 2)
                height1 = 2;
            label2[i].Size    =    new    Size(panel3[i].Width,
label2[i].Height * height1);
            label2[i].Left = (panel3[i].Width - label2[i].Width)
/ 2;
            label2[i].Top = label1[i].Bottom + 5;
```

```
                label3[i] = new Label();
                if (maxCost[i] == avgCost[i] && avgCost[i] ==
minCost[i]) {
                    label3[i].Text = "Вартість: " + maxCost[i] +
"грн.";
                }
                else
                    label3[i].Text = "Вартість: " + "Макс: " +
maxCost[i] + "грн;" + "\n" + "Середня: " + avgCost[i] + "грн; Мін:
" + minCost[i] + "грн.";
                label3[i].TextAlign                                =
ContentAlignment.MiddleCenter;
                label3[i].Font    =    new    Font("Arial",    11,
FontStyle.Regular);
                label3[i].Size                                     =
TextRenderer.MeasureText(label3[i].Text + ".", label3[i].Font);
                label3[i].Left = (panel3[i].Width - label3[i].Width)
/ 2;
                label3[i].Top = label2[i].Bottom + 5;

                label4[i] = new Label();
                label4[i].Text = otzivStr[i];
                label4[i].TextAlign                                =
ContentAlignment.MiddleCenter;
                label4[i].Font    =    new    Font("Arial",    11,
FontStyle.Regular);
                label4[i].Size                                     =
TextRenderer.MeasureText(label4[i].Text + ".", label4[i].Font);
                label4[i].Left = (panel3[i].Width - label4[i].Width)
/ 2;
                label4[i].Top = label3[i].Bottom + 5;

                panelStr1.Controls.Add(panel3[i]);
                panel3[i].Controls.Add(label1[i]);
                panel3[i].Controls.Add(label2[i]);
                panel3[i].Controls.Add(label3[i]);
                panel3[i].Controls.Add(label4[i]);
                panel3[i].Controls.Add(pictureBox1[i]);

                panel3[i].DoubleClick += doublbCli;
                panel3[i].TabIndex = sortedId[i];
            }
            panelStr1.AutoScroll = true;
        }

        private void doublbCli(object sender, EventArgs e)
        {
            var myPanel3 = (Panel)sender;
            if (myPanel3 != null)
            {
                Byte[] byteImg = new Byte[0];
                currentAccess = myPanel3.TabIndex;

                if (currentAccess == curId)
                {
                    trackBar1.Visible = false;
                    otziv_Lable.Visible = false;
                    btn_Otziv.Visible = false;
                    message_Btn.Visible = false;
                }
                else
                {
                    trackBar1.Visible = true;
                    otziv_Lable.Visible = true;
                    btn_Otziv.Visible = true;
                    message_Btn.Visible = true;
```

```
            }
            trackBar1.Value = 1;

            dataBase1.openConnection();

            SqlCommand cmd_InfoTeach = new SqlCommand($"SELECT
Name, SecondName, ThirdName, Description, Favor FROM {MyTable2}
WHERE ID_user = {currentAccess}", dataBase1.getConnection());
            SqlDataAdapter adapter = new SqlDataAdapter();
            DataSet dataSet = new DataSet();
            adapter.SelectCommand = cmd_InfoTeach;
            adapter.Fill(dataSet);
            SqlDataReader                reader                =
cmd_InfoTeach.ExecuteReader();

            string otzivi = "";
            while (reader.Read())
            {
                nameText_Str1.Text = reader.GetString(0);
                secondNameText_Str1.Text = reader.GetString(1);
                thirdNameText_Str1.Text = reader.GetString(2);
                descriptText_Str1.Text = reader.GetString(3);
                otzivi = reader.GetString(4);
            }
            reader.Close();

            String requiredString = otzivi;
            int           countOtziv           =           new
Regex("id:").Matches(otzivi).Count;

            int summ = 0;
            while (requiredString.IndexOf(",") >= 0)
            {
                int firstPos = requiredString.IndexOf(",") + 1;
                int secondPos = requiredString.IndexOf(";");

                summ                                          +=
Convert.ToInt32(requiredString.Substring(firstPos,   secondPos   -
firstPos));
                requiredString                                =
requiredString.Substring(requiredString.IndexOf(";") + 1);
            }

            float avgOtziv = 0;
            if (countOtziv > 0)
            {
                avgOtziv = (float)summ / (float)countOtziv;
                statusOtziv_Lable.Text = "Відгуки(" + countOtziv
+ "): " + avgOtziv.ToString("#.#") + " з 5";
            }
            else {
                statusOtziv_Lable.Text = "Відгуки(" + countOtziv
+ ")";
            }

            string allOtzivi = otzivi;
            string loop = "id:" + curId + ",";
            int indexId = allOtzivi.IndexOf(loop);

            if (indexId == -1)
            {
                otziv_Lable.Text = "Ви ще не залишали відгук";
            }
            else
            {
                allOtzivi = allOtzivi.Remove(0, indexId);
```

```
                indexId = allOtzivi.IndexOf(loop);
                int secondId = allOtzivi.IndexOf(";");

                indexId += loop.Length;
                int myOtziv = Convert.ToInt32(allOtzivi.Substring(indexId, secondId - indexId));
                otziv_Lable.Text = "Ваш відгук: " + myOtziv + " з 5";
                trackBar1.Value = myOtziv;
            }

            SqlCommand cmd_getImg = new SqlCommand($"SELECT Image FROM {MyTable2} WHERE ID_user = {currentAccess}", dataBase1.getConnection());
            SqlDataAdapter da = new SqlDataAdapter(cmd_getImg);
            DataSet ds = new DataSet();
            da.Fill(ds, MyTable2);
            int c = ds.Tables[MyTable2].Rows.Count;

            if (c > 0)
            {
                byteImg = (Byte[])(ds.Tables[MyTable2].Rows[c - 1]["Image"]);
            }

            MemoryStream stmBLOBData = new MemoryStream(byteImg);
            pictureBox_Str1.Image = Image.FromStream(stmBLOBData);

            string sql_CountSubj = $"SELECT COUNT(ID_Subj) FROM {MyTable4} WHERE ID_user = {currentAccess}";
            SqlCommand cmd_CountSubj = new SqlCommand(sql_CountSubj, dataBase1.getConnection());
            int countSubj = (int)cmd_CountSubj.ExecuteScalar();

            string[] subjName = new string[countSubj];
            string[] staj = new string[countSubj];
            string[] deskripSubj = new string[countSubj];
            int[] typeTeach = new int[countSubj];
            int[] cost = new int[countSubj];

            string sql_SubjTable = $"SELECT Subj_Name, TypeTeach, Cost, Staj, Deskription FROM {MyTable4} WHERE ID_user = {currentAccess}";
            SqlCommand cmd_SubjTable = new SqlCommand(sql_SubjTable, dataBase1.getConnection());
            SqlDataAdapter adapter2 = new SqlDataAdapter();
            DataSet dataSet2 = new DataSet();
            adapter2.SelectCommand = cmd_SubjTable;
            adapter2.Fill(dataSet2);
            SqlDataReader reader2 = cmd_SubjTable.ExecuteReader();

            int k = 0;
            while (reader2.Read())
            {
                subjName[k] = reader2.GetString(0);
                typeTeach[k] = reader2.GetInt32(1);
                cost[k] = reader2.GetInt32(2);
                staj[k] = reader2.GetString(3);
                deskripSubj[k] = reader2.GetString(4);
                k++;
            }
            reader2.Close();
```

```
                dataBase1.closeConnection();

                dataGridView2.Rows.Clear();
                int rows = dataGridView2.Rows.Count;

                if (rows <= countSubj)
                {
                    for (int i = 0; i < countSubj; i++)
                    {
                        dataGridView2.Rows.Add();
                    }
                }

                for (int i = 0; i < countSubj; i++)
                {
                    dataGridView2.Rows[i].Cells[0].Value          =
subjName[i];
                    if (typeTeach[i] == 1)
                    {
                        dataGridView2.Rows[i].Cells[1].Value      =
subjIndivArr[0]; //"Групові";
                    }
                    else if (typeTeach[i] == 2)
                    {
                        dataGridView2.Rows[i].Cells[1].Value      =
subjIndivArr[1]; //"Індивідуальні";
                    }

                    dataGridView2.Rows[i].Cells[2].Value = cost[i];
                    dataGridView2.Rows[i].Cells[3].Value = staj[i];
                    dataGridView2.Rows[i].Cells[4].Value          =
deskripSubj[i];
                }
                panelStr1_1.Visible = true;
                panelStr1.Visible = false;
            }
        }


        private void btn_Otziv_Click(object sender, EventArgs e)
        {
            if (currentAccess != curId)
            {
                dataBase1.openConnection();

                string   sql_FindOrziv   =   $"SELECT   Favor   FROM
{MyTable2} WHERE ID_user = {currentAccess}";
                SqlCommand        cmd_FindOrziv        =        new
SqlCommand(sql_FindOrziv, dataBase1.getConnection());
                String                  findOrziv                 =
Convert.ToString(cmd_FindOrziv.ExecuteScalar());
                int indexId = findOrziv.IndexOf("id:" + curId);

                if (indexId == -1)
                {
                    string addOtziv = Convert.ToString(findOrziv +
"id:" + curId + "," + trackBar1.Value + ";");
                    string sql_UpdateOtziv = $"UPDATE {MyTable2} SET
Favor = '{addOtziv}' WHERE ID_user = {currentAccess}";
                    SqlCommand      cmd_UpdateOtziv      =      new
SqlCommand(sql_UpdateOtziv, dataBase1.getConnection());
                    cmd_UpdateOtziv.ExecuteNonQuery();
                }
                else if (indexId >= 0)
                {
```

```
                int indexEnd = findOrziv.IndexOf(";", indexId +
1);
                findOrziv = findOrziv.Remove(indexId, indexEnd -
indexId + 1);
                string updOtziv = Convert.ToString(findOrziv +
"id:" + curId + "," + trackBar1.Value + ";");

                string sql_UpdateOtziv = $"UPDATE {MyTable2} SET
Favor = '{updOtziv}' WHERE ID_user = {currentAccess}";
                SqlCommand      cmd_UpdateOtziv      =      new
SqlCommand(sql_UpdateOtziv, dataBase1.getConnection());
                cmd_UpdateOtziv.ExecuteNonQuery();
            }

            dataBase1.closeConnection();
        }
        else {
            MessageBox.Show("Ви не можете поставити собі від-
гук!", "", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }

    private void panelStr1_Paint(object sender, PaintEventArgs
e)
    {
        if (activeStr1 == 0)
        {
            activeStr1 = 1;
            activeStr2 = 0;
            activeStr3 = 0;
            panelStr1.Controls.Clear();
            Form2_Load(sender, e);
        }
    }

    private void panelStr2_Paint(object sender, PaintEventArgs
e)
    {
        if (activeStr2 == 0)
        {
            activeStr1 = 0;
            activeStr2 = 1;
            activeStr3 = 0;
            panelStr2.Controls.Clear();
            Form2_Load(sender, e);
        }
    }

    private void panelStr3_Paint(object sender, PaintEventArgs
e)
    {
        if (activeStr3 == 0)
        {
            activeStr1 = 0;
            activeStr2 = 0;
            activeStr3 = 1;
            panelStr3.Controls.Clear();
            Form2_Load(sender, e);
        }
    }

    private void backBtn_Str1_Click(object sender, EventArgs e)
    {
        panelStr1.Controls.Clear();
        Form2_Load(sender, e);
        panelStr1_1.Visible = false;
```

```
            panelStr1.Visible = true;
        }

        private   void   studStatus_Box_SelectedIndexChanged(object
sender, EventArgs e)
        {
            studGrade_Box.Text = "";
            studGrade_Box.SelectedIndex = -1;

            if (studStatus_Box.SelectedIndex == 0)
            {
                for (int i = 0; i < studGrade7Arr.Length; i++)
                {
                    studGrade_Box.Items.Add(studGrade7Arr[i]);
                }
            }
            else if (studStatus_Box.SelectedIndex == 1)
            {
                for (int i = 0; i < studGrade11Arr.Length; i++)
                {
                    studGrade_Box.Items.Add(studGrade11Arr[i]);
                }
            }
            else if (studStatus_Box.SelectedIndex == 2)
            {
                for (int i = 0; i < studGradeKursArr.Length; i++)
                {
                    studGrade_Box.Items.Add(studGradeKursArr[i]);
                }
            }
        }

        private void message_Btn_Click(object sender, EventArgs e)
        {
            showMsgOnProfile();
        }

        private  void  backBtn_Str1_2_Click(object  sender,  EventArgs
e)
        {
            panelStr1_1.Visible = true;
            panelStr1_2.Visible = false;
        }

        private void showMsgOnProfile() {
            panelStr1_2_2.Controls.Clear();
            panelStr1_1.Visible = false;
            panelStr1.Visible = false;
            panelStr1_2.Visible = true;

            if (myIdAccess == 0)
                MessageBox.Show("Заповніть усі поля для можливості
відправити    повідомлення!",    "Ок!",    MessageBoxButtons.OK,
MessageBoxIcon.Information);
            else if (myIdAccess == 1)
            {
                int countSendMsg = 0;
                String textMsg = "", lastTime;

                dataBase1.openConnection();

                string sql_CountMessage = $"SELECT COUNT(Id_message)
FROM {MyTable5} WHERE (Id_First = {currentAccess} and Id_Second =
{curId}) or (Id_First = {curId} and Id_Second = {currentAccess})";
                SqlCommand       cmd_CountMessage       =       new
SqlCommand(sql_CountMessage, dataBase1.getConnection());
```

```
            countMyMessage                                          =
(int)cmd_CountMessage.ExecuteScalar();

            if (countMyMessage > 0)
            {
                string  sql_GetMessage  =  $"SELECT  Id_message,
Id_First, Id_Second, Text_Message, Last_Time, Count_Message FROM
{MyTable5}  WHERE  (Id_First  =  {currentAccess}  and  Id_Second  =
{curId}) or (Id_First = {curId} and Id_Second = {currentAccess})";
                SqlCommand      cmd_GetMessage      =      new
SqlCommand(sql_GetMessage, dataBase1.getConnection());
                SqlDataAdapter      adapter_Msg      =      new
SqlDataAdapter();
                DataSet dataSet_Msg = new DataSet();
                adapter_Msg.SelectCommand = cmd_GetMessage;
                adapter_Msg.Fill(dataSet_Msg);
                SqlDataReader            reader_Msg            =
cmd_GetMessage.ExecuteReader();

                while (reader_Msg.Read())
                {
                    textMsg = reader_Msg.GetString(3);
                    lastTime = reader_Msg.GetString(4);
                    countSendMsg = reader_Msg.GetInt32(5);
                }
                reader_Msg.Close();

                Panel[] panelMsg = new Panel[countSendMsg];
                Label[] labelMsg = new Label[countSendMsg];
                Label[] labelDate = new Label[countSendMsg];
                panelStr1_2_2.AutoScroll = true;

                int i = 0;
                String changeMsg = textMsg;

                while (changeMsg.IndexOf("|") >= 0)
                {
                    int firstPos = changeMsg.IndexOf("[IdSend=")
+ 8;
                    int secondPos = changeMsg.IndexOf("|");
                    int                 idUser                  =
Convert.ToInt32(changeMsg.Substring(firstPos,     secondPos     -
firstPos));

                    firstPos = changeMsg.IndexOf("|") + 1;
                    secondPos = changeMsg.IndexOf("~");

                    String               msgText               =
changeMsg.Substring(firstPos, secondPos - firstPos);
                    firstPos = changeMsg.IndexOf("~") + 1;
                    secondPos = changeMsg.IndexOf("]");

                    String               msgDate               =
changeMsg.Substring(firstPos, secondPos - firstPos);
                    changeMsg                                   =
changeMsg.Substring(changeMsg.IndexOf("]") + 1);

                    panelMsg[i] = new Panel();
                    panelMsg[i].MaximumSize        =        new
Size(panelStr1_2_2.Width / 2, panelStr1_2_2.Height - 40);

                    labelMsg[i] = new Label();
                    labelMsg[i].Text = msgText;
                    labelMsg[i].TextAlign                       =
ContentAlignment.MiddleCenter;
```

```
                        labelMsg[i].Font = new Font("Arial", 10,
FontStyle.Regular);
                        labelMsg[i].Size                        =
TextRenderer.MeasureText(labelMsg[i].Text         +         ".",
labelMsg[i].Font);
                        int height1 = (int)((int)(labelMsg[i].Width
/ (panelStr1_2_2.Width / 2 - 20)) + 1);
                        if (height1 > 1)
                        {
                            labelMsg[i].Size         =         new
Size(panelStr1_2_2.Width / 2 - 20, labelMsg[i].Height * height1);
                        }
                        labelMsg[i].Top = 10;

                        labelDate[i] = new Label();
                        labelDate[i].Text = msgDate;
                        labelDate[i].TextAlign                  =
ContentAlignment.MiddleCenter;
                        labelDate[i].Font = new Font("Arial", 8,
FontStyle.Regular);
                        labelDate[i].MaximumSize        =        new
Size(panelMsg[i].Width     -     10,     panelMsg[i].Height     -
labelDate[i].Height - 10);
                        labelDate[i].Size                       =
TextRenderer.MeasureText(labelDate[i].Text         +         ".",
labelDate[i].Font);
                        labelDate[i].Top = labelMsg[i].Bottom + 10;

                        if (labelMsg[i].Width < labelDate[i].Width)
                        {
                            panelMsg[i].Size         =         new
Size(labelDate[i].Width     +     20,     labelMsg[i].Height     +
labelDate[i].Height + 30);
                        }
                        else
                        {
                            panelMsg[i].Size         =         new
Size(labelMsg[i].Width     +     20,     labelMsg[i].Height     +
labelDate[i].Height + 30);
                        }

                        panelMsg[i].BorderStyle                 =
System.Windows.Forms.BorderStyle.Fixed3D;

                        panelMsg[i].Left = 100;
                        if (idUser == curId)
                        {
                            panelMsg[i].Left = (panelStr1_2_2.Width
- panelMsg[i].Width) - 100;
                            panelMsg[i].BackColor               =
Color.FromArgb(55, 71, 79);
                            labelMsg[i].ForeColor = Color.White;
                            labelDate[i].ForeColor = Color.White;
                        }
                        if (i == 0)
                        {
                            panelMsg[i].Top = 30;
                        }
                        else
                        {
                            panelMsg[i].Top = panelMsg[i - 1].Top +
panelMsg[i - 1].Height + 15;
                        }

                        labelMsg[i].Left = 10;
```

```
                    labelDate[i].Left = (panelMsg[i].Width -
labelDate[i].Width) / 2;

                    panelStr1_2_2.Controls.Add(panelMsg[i]);
                    panelMsg[i].Controls.Add(labelMsg[i]);
                    panelMsg[i].Controls.Add(labelDate[i]);
                    i++;
                }
                panelStr1_2_2.AutoScrollPosition = new Point(0,
panelMsg[i - 1].Top + panelMsg[i - 1].Height + 50);
            }
            dataBase1.closeConnection();
        }
    }

    private void sendMsg_Btn_Click(object sender, EventArgs e)
    {
        if (message_Box.Text != "")
        {
            string last_Time = DateTime.Now.ToShortDateString()
+ " " + DateTime.Now.ToShortTimeString();
            string text_Message = "[IdSend=" + curId + "|" +
message_Box.Text + "~" + last_Time + "]";

            dataBase1.openConnection();

            string sql_CountMessage = $"SELECT Count_Message
FROM {MyTable5} WHERE (Id_First = {currentAccess} and Id_Second =
{curId}) or (Id_First = {curId} and Id_Second = {currentAccess})";
            SqlCommand cmd_CountMessage = new
SqlCommand(sql_CountMessage, dataBase1.getConnection());
            int countMessage =
Convert.ToInt32(cmd_CountMessage.ExecuteScalar());

            if (countMessage <= 0)
            {
                string sql_AddMsg = $"INSERT into {MyTable5}
(Id_First, Id_Second, Text_Message, Last_Time, Count_Message)
values ({curId}, {currentAccess}, '{text_Message}', '{last_Time}',
{countMessage + 1})";
                SqlCommand cmd_AddMsg = new
SqlCommand(sql_AddMsg, dataBase1.getConnection());
                cmd_AddMsg.ExecuteNonQuery();
            }
            else if (countMessage > 0)
            {
                string sql_TextMessage = $"SELECT Text_Message
FROM {MyTable5} WHERE (Id_First = {currentAccess} and Id_Second =
{curId}) or (Id_First = {curId} and Id_Second = {currentAccess})";
                SqlCommand cmd_TextMessage = new
SqlCommand(sql_TextMessage, dataBase1.getConnection());
                string allTextMessage =
Convert.ToString(cmd_TextMessage.ExecuteScalar());

                string sql_UpdMsg = $"UPDATE {MyTable5} SET
Id_First = {curId}, Id_Second = {currentAccess}, Text_Message =
'{allTextMessage + text_Message}', Last_Time = '{last_Time}',
Count_Message = {countMessage + 1} WHERE (Id_First =
{currentAccess} and Id_Second = {curId}) or (Id_First = {curId}
and Id_Second = {currentAccess})";
                SqlCommand cmd_UpdMsg = new
SqlCommand(sql_UpdMsg, dataBase1.getConnection());
                cmd_UpdMsg.ExecuteNonQuery();
            }
            dataBase1.closeConnection();
            message_Box.Text = "";
```

```
            showMsgOnProfile();
        }
    }

    public void CreateMessagePanel()
    {
        dataBase1.openConnection();
        string sql_CountMessage = $"SELECT COUNT(Id_message)
FROM {MyTable5} WHERE Id_First = {curId} or Id_Second = {curId}";
        SqlCommand cmd_CountMessage = new
SqlCommand(sql_CountMessage, dataBase1.getConnection());
        int countMsgPanel =
Convert.ToInt32(cmd_CountMessage.ExecuteScalar());
        int[] teachID = new int[countMsgPanel];

        SqlCommand cmd_Msg = new SqlCommand($"SELECT Id_message,
Id_First, Id_Second, Last_Time FROM {MyTable5} WHERE Id_First =
{curId} or Id_Second = {curId}", dataBase1.getConnection());
        SqlDataAdapter adapterMsg = new SqlDataAdapter();
        DataSet dataSetMsg = new DataSet();
        adapterMsg.SelectCommand = cmd_Msg;
        adapterMsg.Fill(dataSetMsg);
        SqlDataReader readerMsg = cmd_Msg.ExecuteReader();

        int[] idMsg = new int[countMsgPanel];
        string[] lastTime = new string[countMsgPanel];
        DateTime[] parsedDate = new DateTime[countMsgPanel];

        int k = 0;
        while (readerMsg.Read())
        {
            idMsg[k] = readerMsg.GetInt32(0);
            int idFirst = readerMsg.GetInt32(1);
            int idSecond = readerMsg.GetInt32(2);
            lastTime[k] = readerMsg.GetString(3);
            if (idFirst != curId)
                teachID[k] = idFirst;
            else
                teachID[k] = idSecond;

            parsedDate[k] = DateTime.Parse(lastTime[k]);
            k++;
        }
        readerMsg.Close();
        dataBase1.closeConnection();

        Panel[] panelMsg = new Panel[countMsgPanel];
        Label[] label1 = new Label[countMsgPanel]; // ПІБ кому
писали
        Label[] label2 = new Label[countMsgPanel]; // останнє
повідомлення
        string[] FIO = new string[countMsgPanel];

        dataBase1.openConnection();
        for (int i = 0; i < countMsgPanel; i++)
        {
            string sql_CountTeach = $"SELECT COUNT(ID_user) FROM
{MyTable2} WHERE ID_user = {teachID[i]}";
            SqlCommand cmd_CountTeach = new
SqlCommand(sql_CountTeach, dataBase1.getConnection());
            int countTeach =
Convert.ToInt32(cmd_CountTeach.ExecuteScalar());

            if (countTeach > 0)
            {
```

```
                SqlCommand      cmd_NameTeach      =      new
SqlCommand($"SELECT Name, SecondName, ThirdName FROM {MyTable2}
WHERE ID_user = {teachID[i]}", dataBase1.getConnection());
                SqlDataAdapter adapter5 = new SqlDataAdapter();
                DataSet dataSet5 = new DataSet();
                adapter5.SelectCommand = cmd_NameTeach;
                adapter5.Fill(dataSet5);
                SqlDataReader             reader5             =
cmd_NameTeach.ExecuteReader();

                while (reader5.Read())
                {
                    FIO[i]  =  reader5.GetString(0)  +  "  "  +
reader5.GetString(1) + " " + reader5.GetString(2);
                }
                reader5.Close();
            }
            else {
                SqlCommand      cmd_NameTeach      =      new
SqlCommand($"SELECT Name, SecondName, ThirdName FROM {MyTable3}
WHERE ID_user = {teachID[i]}", dataBase1.getConnection());
                SqlDataAdapter adapter5 = new SqlDataAdapter();
                DataSet dataSet5 = new DataSet();

                adapter5.SelectCommand = cmd_NameTeach;
                adapter5.Fill(dataSet5);

                SqlDataReader             reader5             =
cmd_NameTeach.ExecuteReader();

                while (reader5.Read())
                {
                    FIO[i]  =  reader5.GetString(0)  +  "  "  +
reader5.GetString(1) + " " + reader5.GetString(2);
                }
                reader5.Close();
            }

            panelMsg[i] = new Panel();
            if (i == 0)
            {
                panelMsg[i].Top = 10;
            }
            else
                panelMsg[i].Top = panelMsg[i-1].Top + panelMsg[i
- 1].Height + 10;

            panelMsg[i].Left = 10;
            panelMsg[i].Size   =   new   Size(panelStr2.Width-20,
100);
            panelMsg[i].BorderStyle                            =
System.Windows.Forms.BorderStyle.Fixed3D;
            panelMsg[i].BackColor = Color.FromArgb(55, 71, 79);

            label1[i] = new Label();
            label1[i].Text = FIO[i];
            label1[i].TextAlign                                =
ContentAlignment.MiddleCenter;
            label1[i].Font    =    new    Font("Arial",    11,
FontStyle.Regular);
            label1[i].MaximumSize = new Size(panelMsg[i].Width,
label1[i].Height);
            label1[i].Size                                     =
TextRenderer.MeasureText(label1[i].Text + ".", label1[i].Font);
            label1[i].Left = panelMsg[i].Width/5;
```

```
            label1[i].Top      =      (panelMsg[i].Height      -
label1[i].Height) / 2;
            label1[i].ForeColor = Color.White;

            label2[i] = new Label();
            label2[i].Text   =   "Останнє   повідомлення:   "   +
lastTime[i];
            label2[i].TextAlign                                  =
ContentAlignment.MiddleCenter;
            label2[i].Font    =    new    Font("Arial",    11,
FontStyle.Regular);
            label2[i].Size                                       =
TextRenderer.MeasureText(label2[i].Text + ".", label2[i].Font);
            label2[i].Left = (int)(panelMsg[i].Width * 0.6);
            label2[i].Top      =      (panelMsg[i].Height      -
label2[i].Height) / 2;
            label2[i].ForeColor = Color.White;

            panelStr2.Controls.Add(panelMsg[i]);
            panelMsg[i].Controls.Add(label1[i]);
            panelMsg[i].Controls.Add(label2[i]);

            panelMsg[i].DoubleClick += doubleCli_Msg;
            panelMsg[i].TabIndex = teachID[i];
        }
        panelStr2.AutoScroll = true;
    }

    private void doubleCli_Msg(object sender, EventArgs e)
    {
        var panelMsg = (Panel)sender;

        if (panelMsg != null)
        {
            teachIdMsg = panelMsg.TabIndex;
        }
        messageShow();
        panelMsgMain.Visible = true;
        panelStr2.Visible = false;
    }

    private void backMsg_Btn_Click(object sender, EventArgs e)
    {
        panelMsgMain.Visible = false;
        panelStr2.Visible = true;
    }

    private void buttonMsg_Click(object sender, EventArgs e)
    {
        var btnMsg = (Button)sender;

        if (btnMsg != null)
        {
            teachIdMsg = btnMsg.TabIndex;
        }

        if (textMsgBox.Text != "")
        {
            string last_Time = DateTime.Now.ToShortDateString()
+ " " + DateTime.Now.ToShortTimeString();
            string text_Message = "[IdSend=" + curId + "|" +
textMsgBox.Text + "~" + last_Time + "]";

            dataBase1.openConnection();
```

```
            string sql_CountMessage = $"SELECT Count_Message
FROM {MyTable5} WHERE (Id_First = {teachIdMsg} and Id_Second =
{curId}) or (Id_First = {curId} and Id_Second = {teachIdMsg})";
            SqlCommand cmd_CountMessage = new
SqlCommand(sql_CountMessage, dataBase1.getConnection());
            int countMessage =
Convert.ToInt32(cmd_CountMessage.ExecuteScalar());

            if (countMessage <= 0)
            {
                string sql_AddMsg = $"INSERT into {MyTable5}
(Id_First, Id_Second, Text_Message, Last_Time, Count_Message)
values ({curId}, {teachIdMsg}, '{text_Message}', '{last_Time}',
{countMessage + 1})";
                SqlCommand cmd_AddMsg = new
SqlCommand(sql_AddMsg, dataBase1.getConnection());
                cmd_AddMsg.ExecuteNonQuery();
            }
            else if (countMessage > 0)
            {
                string sql_TextMessage = $"SELECT Text_Message
FROM {MyTable5} WHERE (Id_First = {teachIdMsg} and Id_Second =
{curId}) or (Id_First = {curId} and Id_Second = {teachIdMsg})";
                SqlCommand cmd_TextMessage = new
SqlCommand(sql_TextMessage, dataBase1.getConnection());
                string allTextMessage =
Convert.ToString(cmd_TextMessage.ExecuteScalar());

                string sql_UpdMsg = $"UPDATE {MyTable5} SET
Id_First = {curId}, Id_Second = {teachIdMsg}, Text_Message =
'{allTextMessage + text_Message}', Last_Time = '{last_Time}',
Count_Message = {countMessage + 1} WHERE (Id_First = {teachIdMsg}
and Id_Second = {curId}) or (Id_First = {curId} and Id_Second =
{teachIdMsg})";
                SqlCommand cmd_UpdMsg = new
SqlCommand(sql_UpdMsg, dataBase1.getConnection());
                cmd_UpdMsg.ExecuteNonQuery();
            }

            dataBase1.closeConnection();

            textMsgBox.Text = "";
            panelMsgMain.Visible = false;
            tabPage2.Controls.Remove(panelMsgMain);
            messageShow();

            panelMsgMain.Visible = true;
            panelStr2.Visible = false;
        }
    }

    private void messageShow() {

        if (myIdAccess == 0)
            MessageBox.Show("Заповніть усі поля для можливості
відправити повідомлення!", "Ок!", MessageBoxButtons.OK,
MessageBoxIcon.Information);
        else if (myIdAccess == 1)
        {
            int idMsg = -1, idFirst = -1, idSecond = -1,
countSendMsg = 0;
            String textMsg = "", lastTime;

            dataBase1.openConnection();
```

```
            string sql_CountMessage = $"SELECT COUNT(Id_message)
FROM {MyTable5} WHERE (Id_First = {teachIdMsg} and Id_Second =
{curId}) or (Id_First = {curId} and Id_Second = {teachIdMsg})";
            SqlCommand       cmd_CountMessage       =       new
SqlCommand(sql_CountMessage, dataBase1.getConnection());
            countMyMessage                                    =
(int)cmd_CountMessage.ExecuteScalar();

            string   sql_GetMessage   =   $"SELECT   Id_message,
Id_First, Id_Second, Text_Message, Last_Time, Count_Message FROM
{MyTable5} WHERE (Id_First = {teachIdMsg} and Id_Second = {curId})
or (Id_First = {curId} and Id_Second = {teachIdMsg})";
            SqlCommand       cmd_GetMessage       =       new
SqlCommand(sql_GetMessage, dataBase1.getConnection());
            SqlDataAdapter adapter_Msg = new SqlDataAdapter();
            DataSet dataSet_Msg = new DataSet();

            adapter_Msg.SelectCommand = cmd_GetMessage;
            adapter_Msg.Fill(dataSet_Msg);

            SqlDataReader              reader_Msg              =
cmd_GetMessage.ExecuteReader();

            while (reader_Msg.Read())
            {
                idMsg = reader_Msg.GetInt32(0);
                idFirst = reader_Msg.GetInt32(1);
                idSecond = reader_Msg.GetInt32(2);
                textMsg = reader_Msg.GetString(3);
                lastTime = reader_Msg.GetString(4);
                countSendMsg = reader_Msg.GetInt32(5);
            }
            reader_Msg.Close();

            String readTextMsg = textMsg;
            int            countMsg1            =            new
Regex("id:").Matches(textMsg).Count;

            Panel panelMsg1 = new Panel();
            Panel panelMsg2 = new Panel();
            Panel panelMsg3 = new Panel();
            Panel[] panelMsg4 = new Panel[countSendMsg];
            Label[] labelMsg = new Label[countSendMsg];
            Label[] labelDate = new Label[countSendMsg];
            Button buttonMsg = new Button();
            MaterialRaisedButton     backMsg_Btn     =     new
MaterialRaisedButton();

            panelMsgMain = new Panel();
            panelMsgMain.Location = new Point(0, 0);
            panelMsgMain.Size   =   new   Size(tabPage2.Width,
tabPage2.Height);

            panelMsg1.Location = new Point(0, 0);
            panelMsg1.Size = new Size(panelMsgMain.Width, 60);
            panelMsg1.BackColor = Color.Silver;

            backMsg_Btn.Location = new Point(15, 15);
            backMsg_Btn.Text = "<";
            backMsg_Btn.Size = new Size(38, 30);

            panelMsg2.Location = new Point(0, panelMsg1.Height);
            panelMsg2.Size = new Size(panelMsgMain.Width, 540);
            panelMsg2.AutoScroll = true;
```

```
            panelMsg3.Location = new Point(0, panelMsg1.Height +
panelMsg2.Height);
            panelMsg3.Size = new Size(panelMsgMain.Width, 125);
            panelMsg3.BackColor = Color.FromArgb(55, 71, 79);

            textMsgBox.Size = new Size(580, 95);
            textMsgBox.Location = new Point((panelMsg3.Width -
textMsgBox.Width) / 2, (panelMsg3.Height - textMsgBox.Height) /
2);

            buttonMsg.Text = "Відправити";
            buttonMsg.Size = new Size(75, 35);
            buttonMsg.Location = new Point(textMsgBox.Left +
textMsgBox.Width + 120, (panelMsg3.Height - buttonMsg.Height) /
2);
            buttonMsg.BackColor = Color.Silver;

            tabPage2.Controls.Add(panelMsgMain);
            panelMsgMain.Controls.Add(panelMsg1);
            panelMsg1.Controls.Add(backMsg_Btn);
            panelMsgMain.Controls.Add(panelMsg2);
            panelMsgMain.Controls.Add(panelMsg3);
            panelMsg3.Controls.Add(textMsgBox);
            panelMsg3.Controls.Add(buttonMsg);

            backMsg_Btn.Click              +=              new
EventHandler(backMsg_Btn_Click);
            buttonMsg.Click               +=               new
EventHandler(buttonMsg_Click);
            buttonMsg.TabIndex = teachIdMsg;
            textMsgBox.TabIndex = 1000;

            int i = 0;
            String changeMsg = textMsg;

            while (changeMsg.IndexOf("|") >= 0)
            {
                int firstPos = changeMsg.IndexOf("[IdSend=") +
8;
                int secondPos = changeMsg.IndexOf("|");

                int                  idUser                  =
Convert.ToInt32(changeMsg.Substring(firstPos,     secondPos     -
firstPos));
                firstPos = changeMsg.IndexOf("|") + 1;
                secondPos = changeMsg.IndexOf("~");

                String msgText = changeMsg.Substring(firstPos,
secondPos - firstPos);
                firstPos = changeMsg.IndexOf("~") + 1;
                secondPos = changeMsg.IndexOf("]");

                String msgDate = changeMsg.Substring(firstPos,
secondPos - firstPos);
                changeMsg                                      =
changeMsg.Substring(changeMsg.IndexOf("]") + 1);

                panelMsg4[i] = new Panel();
                panelMsg4[i].MaximumSize          =          new
Size(panelMsg2.Width / 2, panelMsg2.Height - 40);

                labelMsg[i] = new Label();
                labelMsg[i].Text = msgText;
                labelMsg[i].TextAlign                          =
ContentAlignment.MiddleCenter;
```

```
                labelMsg[i].Font   =   new   Font("Arial",   10,
FontStyle.Regular);

                labelMsg[i].Size                                  =
TextRenderer.MeasureText(labelMsg[i].Text         +         ".",
labelMsg[i].Font);
                int  height1  =  (int)((int)(labelMsg[i].Width  /
(panelMsg2.Width / 2 - 20)) + 1);
                if (height1 > 1)
                {
                    labelMsg[i].Size = new Size(panelMsg2.Width
/ 2 - 20, labelMsg[i].Height * height1);
                }
                labelMsg[i].Top = 10;

                labelDate[i] = new Label();
                labelDate[i].Text = msgDate;
                labelDate[i].TextAlign                            =
ContentAlignment.MiddleCenter;
                labelDate[i].Font   =   new   Font("Arial",   8,
FontStyle.Regular);
                labelDate[i].MaximumSize          =          new
Size(panelMsg4[i].Width     -     10,     panelMsg4[i].Height     -
labelDate[i].Height - 10);
                labelDate[i].Size                                 =
TextRenderer.MeasureText(labelDate[i].Text         +         ".",
labelDate[i].Font);
                labelDate[i].Top = labelMsg[i].Bottom + 10;

                if (labelMsg[i].Width < labelDate[i].Width)
                {
                    panelMsg4[i].Size           =           new
Size(labelDate[i].Width     +     20,     labelMsg[i].Height     +
labelDate[i].Height + 30);
                }
                else
                {
                    panelMsg4[i].Size           =           new
Size(labelMsg[i].Width     +     20,     labelMsg[i].Height     +
labelDate[i].Height + 30);
                }

                panelMsg4[i].BorderStyle                          =
System.Windows.Forms.BorderStyle.Fixed3D;

                panelMsg4[i].Left = 100;
                if (idUser == curId)
                {
                    panelMsg4[i].Left   =   (panelMsg2.Width   -
panelMsg4[i].Width) - 100;
                    panelMsg4[i].BackColor = Color.FromArgb(55,
71, 79);
                    labelMsg[i].ForeColor = Color.White;
                    labelDate[i].ForeColor = Color.White;
                }

                if (i == 0)
                {
                    panelMsg4[i].Top = 30;
                }
                else
                {
                    panelMsg4[i].Top  =  panelMsg4[i  -  1].Top  +
panelMsg4[i - 1].Height + 15;
                }
```

```
                labelMsg[i].Left = 10;
                labelDate[i].Left   =   (panelMsg4[i].Width   -
labelDate[i].Width) / 2;

                panelMsg2.Controls.Add(panelMsg4[i]);
                panelMsg4[i].Controls.Add(labelMsg[i]);
                panelMsg4[i].Controls.Add(labelDate[i]);
                i++;
            }
            panelMsg2.AutoScrollPosition    =    new    Point(0,
panelMsg4[i-1].Top + panelMsg4[i-1].Height + 50);

            dataBase1.closeConnection();
        }
    }

    private void AddPhoto_Btn_Click(object sender, EventArgs e)
    {
        OpenFileDialog OPF = new OpenFileDialog();
        OPF.Filter = "Файлы jpg|*.jpg|Файлы png|*.png";
        if (OPF.ShowDialog() == DialogResult.OK)
        {
            MessageBox.Show(OPF.FileName);
        }

        if (OPF.FileName != "")
        {
            dataBase1.openConnection();
            String strBLOBFilePath = $@"{OPF.FileName}";
            FileStream         fsBLOBFile         =         new
FileStream(strBLOBFilePath, FileMode.Open, FileAccess.Read);

            Byte[]         bytBLOBData         =         new
Byte[fsBLOBFile.Length];
            fsBLOBFile.Read(bytBLOBData,                  0,
bytBLOBData.Length);
            fsBLOBFile.Close();

            String sql_regCard;
            if (status == 1)
            {
                sql_regCard = $"UPDATE {MyTable2} SET Image
= @Img WHERE ID_user = {curId}";
            }
            else
            {
                sql_regCard = $"UPDATE {MyTable3} SET Image
= @Img WHERE ID_user = {curId}";
            }

            SqlCommand        cmd_regCard        =        new
SqlCommand(sql_regCard, dataBase1.getConnection());
            SqlParameter  prm  =  new  SqlParameter("@Img",
SqlDbType.VarBinary,                          bytBLOBData.Length,
ParameterDirection.Input,      false,      0,      0,      null,
DataRowVersion.Current, bytBLOBData);
            cmd_regCard.Parameters.Add(prm);
            cmd_regCard.ExecuteNonQuery();

            dataBase1.closeConnection();
            Form2_Load(sender, e);
        }
    }

    private void btn_FindCard_Click(object sender, EventArgs e)
    {
```

```
            panelStr1.Controls.Clear();
            Form2_Load(sender, e);
            panelStr1_1.Visible = false;
            panelStr1_2.Visible = false;
            panelStr1.Visible = true;
        }

        private void checkBox_All_CheckedChanged(object sender,
    EventArgs e)
        {
            if (checkBox_All.Checked == true)
            {
                for (int i = 0; i < countObj; i++)
                {
                    checkObj[i].Checked = true;
                }
                for (int i = 0; i < 2; i++)
                {
                    checkIndiv[i].Checked = true;
                }
            }
            else {
                for (int i = 0; i < countObj; i++)
                {
                    checkObj[i].Checked = false;
                }
                for (int i = 0; i < 2; i++)
                {
                    checkIndiv[i].Checked = false;
                }
            }
        }

        private void exit_Btn_Click(object sender, EventArgs e)
        {
            Form1 frm1 = new Form1();
            frm1.Show();
            this.Hide();
        }

    }
}
```