Міністерство освіти і науки України

Національний університет «Запорізька політехніка»

Кафедра програмних засобів

ЗВІТ

Дисципліна «Фреймворки розробки програмного забезпечення»

Робота №8

Тема «Оформлення авторського свідоцтва на розроблену програму»

Виконав варіант 19

Студент КНТ-122 Онищенко О. А.

Прийняли

Викладач Зелік О. В.

2024

Мета: Засвоїти основні поняття про порядок та особливості оформлення авторського свідоцтва в Україні на розроблену програму, навчитися виділяти та представляти отримані результати.

Завдання: Розглянути розроблену програмну систему на предмет відповідності поставленому ТЗ. Оформити настанову щодо використання створеного програмного продукту, що має складатися з наступних елементів:

- Призначення програми;
- Структура системи;
- Функціонування системи;
- Структура та органіація даних;
- Основні функції;
- Інтерфейсні засоби;
- Вимоги до програмного та апаратного забезпечення;
- Методика роботи користувача з системою;

Оформити заяву про реєстрацію авторського права на твір. Оформити договір між роботодавцем та авторами комп'ютерної програми. Створити презентацію для представлення результатів розробленого продукту.

НАСТАНОВА ЩОДО ВИКОРИСТАННЯ

Призначення програми

Програма призначена для купівлі та продажу об'єктів нерухомості.

Структура системи

Система поділена на два рівні: видимий користувачеві (фронтенд) та невидимий користувачеві (бекенд).

Рівень фронтенду реалізовано мовою C# і фреймворком WPF. Елементи інтерфейсу додано використовуючи існуючі віджети WPF зі зміненими стилями.

На рівні бекенду система розподілена на два шари: функції доступні фронтенду (моделі представлення) та класи й методи взаємодії з даними (моделі). Фунції шару моделі представлення використовуються елементами інтерфейсу, а у самих функціях виконується доступ до рівню моделей та допоміжних класів.

Рівень фронтенду:

- LoginPage
- ProfilePage

Рівень бекенду:

- Database
- Estate
- EstateKind
- Meeting
- MeetingScore
- MeetingStatus
- User
- UtilityFunctions
- UtilityVariables

Кожен пункт рівню фронтенду поділено на два файли: типу XAML та типу CS. Файл першого типу визначає вигляд інтерфейсу користувача, файл другого типу виконує роль шару моделі представлення і з’єднує інтерфейс користувача із рівнем моделей.

Функціонування системи

Система починає роботу із запуску сторінку входу. На ній користувач має ввести ім’я та увійти у свій кабінет. Після входу користувачеві доступна сторінка профілю, на якій по вкладках представлені елементи системи.

Вкладки основної сторінки програми:

- Домівка (Home) надає інформація про особисті дані користувача: ім’я, баланс, статус.
- Ринок (Available Estate) показує доступні для придбання оголошення.
- Наявні оголошення (Owned Estate) показує об’єкти нерухомості у наявності користувача.
- Вхідні зустрічі (Incoming Meetings) показує список зустрічей для оголошень користувача.
- Вихідні зустрічі (Outgoing Meetings) виводить список зустічей, призначених користувачем.

Структура та органіація даних

Дані програми зберігаються на базі MySQL та організовані у три таблиці: user, estate, meeting.

Таблиця user має поля:

- name (текстове поле): ім’я користувача
- admin (булеве значення): статус менеджера
- balance (цілочисельне значення): баланс

Таблиця estate має поля:

- owner_id (цілочисельне значення): зовнішній ключ до ідентифікатора користувача власника нерухомості
- title (текстове поле): назва нерухомості

- kind (текстове поле з обмеженим вибором): тип нерухомості, може бути Home (приватний будинок), Flat (квартира у будинку), New (квартира у новобудові)
- price (цілочисельне значення): вартість нерухомості

Таблиця meeting має поля:

- sender_id (цілочисельне значення): зовнішній ключ до ідентифікатора користувача замовника зустрічі
- target_id (цілочисельне значення): зовнішній ключ до ідентифікатора об’єкта нерухомості який переглядається
- status (текстове поле з обмеженим вибором): статус зустрічі, може бути Wait (зустріч чекає на опрацювання власником оголошення), Done (зустріч відбулася), Skip (власник оголошення відмовився від зустрічі)
- score (текстове поле з обмеженим вибором): оцінка користувача переглядача оголошення, може бути Bad (не сподобалося), Okay (нормально), Fine (сподобалося)

Кожен запис усіх таблиць має поле унікального ідентифікатора запису id типу int (ціле число).

Програма підключається до бази даних через доєднувач MySQL і опрацьовує дані у класи моделей: User, Estate, Meeting.

Клас даних User має поля:

- ID (ціле число): унікальний ідентифікатор користувача
- Name (рядок): ім’я користувача
- Admin (ціле число): позначка менеджера, за замовчуванням Нема
- Balance (ціле число): поточний баланс, за замовчуванням 0

Клас даних Estate має поля:

- ID (ціле число): унікальний ідентифікатор об’єкта
- Owner (об’єкт User): користувач власник нерухомості
- Title (рядок): назва

- Kind (рядок): тип
- Price (ціле число): ціна, за замовчуванням 0

Клас даних Meeting має поля:

- ID (ціле число): унікальний ідентифікатор зустрічі
- Sender (об’єкт User): користувач переглядач оголошення
- Target (об’єкт Estate): об’єкт нерухомості на перегляд
- Score (рядок): оцінка зустрічі, за замовчуванням Unrated
- Status (рядок): статус зустрічі, за замовчуванням Wait

Доєднувач MySQL працює з базою даних та використовує класи даних для створення об’єктів з прочитаної інформації. Клас роботи з базою даних Database реалізує базові операції з даними: додавання, читання, редагування, видалення.

Програма також містить статичні класи з константними назвами для полів, які мають обмежений вибір: MeetingScore, EstateKind, MeetingStatus. Такі класи необхідні для уникнення помилок написання константних значень та для увімкнення підказок у редакторі коду. Класи є статичними і не потребують оголошення об’єктів їхнього типу бо всі дані в них є незмінними.

Клас MeetingScore містить такі дані:

- Bad (рядок): значення Bad
- Okay (рядок): значення Okay
- Fine (рядок): значення Fine

Клас MeetingStatus містить такі дані:

- Wait (рядок): значення Wait
- Done (рядок): значення Done
- Skip (рядок): значення Skip

Клас EstateKind містить такі дані:

- Home (рядок): значення Home
- Flat (рядок): значення Flat

- New (рядок): значення New

Програма містить класи для допоміжних функцій та змінних: UtilityFunctions та UtilityVariables.

Клас UtilityFunctions містить методи:

- checkEstateKind(kind): перевірити значення типу нерухомості; приймає введене значення типу як параметр та повертає булевий результат на виході
- checkMeetingStatus(status): перевірити значення статусу зустрічі; приймає введене значення статусу як параметр та повертає булевий результат на виході
- checkMeetingScore(score): перевірити значення оцінки зустрічі; приймає введене значення оцінки як параметр та повертає булевий результат на виході

Клас UtilityVariables містить змінну:

- connectionString (рядок): сформований рядок ідентифікаційної інформації для з’єднання з базою даних

Основні функції

Функції системи умовно поділяються на пов’язані з користувачем, оголошеннями та зустрічами.

Для користувача програма має функції входу до профілю, зміни імені, балансу та статусу.

Для оголошень програма надає можливість купувати оголошення, змінювати наявні об’єкти, продавати нерухомість, призначати зустрічі для оголошень з ринку.

Для зустрічей програма дозволяє призначати нові зустрічі для обраного оголошення, показувати вхідні (такі що надійшли від інших користувачів на перегляд оголошень які користувач має в наявності) та

вихідні зустрічі (такі що користувач надіслав на перегляд оголошень інших), показувати лише очікуючі вхідні зустрічі, змінювати статус вхідних зустрічей, змінювати рейтинг вихідних зустрічей.

Інтерфейсні засоби

Графічний інтерфейс користувача виконано через сторінки для легшої навігації, усім елементам тексту надано шрифт Verdana, усі інші елементи взаємодії виконано з доступних фреймворку WPF та передбачені обробники подій для необхідної взаємодії з даними.

Вимоги до програмного та апаратного забезпечення

Програма вимагає мінімальних програмних характеристик:

- Операційна система: Windows 11, 10, 8.1, 7, Server 2008
- Середовище розробки: Microsoft Visual Studio 2015 / 2017 / 2019 / 2022
- Фреймворк .NET: 4.6.2 – 4.8 / 6 / 8

Програма також вимагає мінімальних апаратних характеристик:

- Процесор: 1 ГГц
- Оперативна пам’ять: 512 МБ
- Дисковий простір: 2 ГБ

Методика роботи користувача з системою

Коли користувач взаємодіє з системою через графічний інтерфейс, програма опрацьовує запити (події) та виконує функції через шар моделі представлення. Методи шару моделі представлення виконують запити до

бази даних через клас моделі Database, отримані дані передають на інтерфейс який виводить оновлену інформацію користувачеві після взаємодії.

Основне меню програми розділене на вкладки, кожна з яких має дані. При переході на нову вкладку дані на ній оновлюються моделлю представлення. При отриманні результатів від бази даних використані засоби LINQ для специфічного фільтрування даних за параметрами для необхідної ситуації.

ПРИКЛАД ЗАЯВИ ПРО РЕЄСТРАЦІЮ АВТОРСЬКОГО ПРАВА

Форма
Національний орган інтелектуальної власності
Державна організація «Український національний офіс інтелектуальної власності та інновацій»
Вул. Дмитра Годзенка, 1, м. Київ-42, 01601

Заява
На державну реєстрацію (присвячення у суспільне надбання) авторського права на твір

Номер заяви *7777777* (не правда, як приклад) Дата подання *02.12.2024*

1. Відомості про твір

1.1 Вид і повна назва твору *Комп'ютерна програма «Застосунок для продажу нерухомості»*

1.4 Дата остаточного завершення роботи над твором: *02.12.2024*

2. Відомство про автора (співавтора) твору

2.1 Прізвище, ім'я, по батькові (за наявності) автора *Онищенко Олег Антонович*

2.3 Інформація про автора

Дата народження: *21 вересня 2005*

РНОКПП *210920057139* (не правда, як приклад)

Задекларованe/зареєстроване місце проживання *Вулиця, 21, Запоріжжя, Україна* (не правда, як приклад)

Унікальний номер запису в Єдиному державноум демографічному реєстрі *3105892349527* (не правда, як приклад)

2.4 Якщо заявка подається автором

Адреса для листування *Жуковського, 60, Запоріжжя, Україна* (не правда, як приклад)

Телефон *+380680077147* Адреса електронної пошти *seesmwork@gmail.com*

2.5 Суть авторства, авторського внеску у створення твору (автор, співавтор, упорядник) *автор*

2.6 Майнові права на твір належать автору *Не належать*

2.7 Твір створено *У зв'язку з виконанням обов'язків за трудовим договором (контрактом)*

2.8 Автор буде зазначатися в офіційному електронному бюлетені «Авторське право і суміжні права» *Анонімно*

5. Форма та порядок отримання свідоцтва

Електронна форма

Прошу надіслати свідоцтво за адресою електронної пошти *seesmwork@gmail.com*

6. Перелік документів і матеріалів, що додаються до заяви:

6.1 Копія твору у відповідній формі вираження на аркуші *21* арк.

ДОГОВІР МІЖ РОБОТОДАВЦЕМ ТА АВТОРОМ

Договір №1 (не правда, як приклад)

Про передання (відчуження) майнових прав

М. Запоріжжя 02 грудня 2024 року

(Прізвище, ім'я, по батькові (П.І.Б.) фізичної особи або повне найменування юридичної особи) *Онищенко Олег Антонович* в особі *автора*; (посада та П.І.Б особи, яка уповноважена укладати договір) *громадськість по всьому світу*, яка діє на підставі

(Статуту, Положення, іншого документа) *закону про Авторські права Стаття 32. Перехід прав на твір у порядку спадкування. Перехід твору у суспільне надбання. Захист особистих немайнових прав автора*, (далі – Набувач), з іншого боку, а разом іменовані – Сторони, уклали цей договір (далі – Договір) про наступне:

1. ВИЗНАЧЕННЯ ТЕРМІНІВ

Терміни, що використовуються в цьому Договорі, означають:
відчужувач – особа, яка на підставі договору передає належні їй майнові права іншій особі, після чого ця особа стає суб'єктом авторського права;
набувач – особа, яка на підставі договору набуває майнових прав та стає суб'єктом авторського права;
повна передача прав - передання майнових прав на твір, встановлених статтею 440 Цивільного кодексу України, частиною першою статті 15 Закону України „Про авторське право і суміжні права", без обмежень способів використання твору, зазначених у статті 441 цього Кодексу, частиною третьою статті 15 цього Закону;
часткова передача прав – передання прав, при якому в договорі про передання майнових прав на твір обмежено способи його використання, визначені у статті 441 Цивільного кодексу України, частині третій статті 15 Закону;
Решта термінів, що використовуються в Договорі, визначаються згідно з нормами чинного законодавства України.

2. ПРЕДМЕТ ДОГОВОРУ

2.1. Відчужувач у повному складі передає (відчужує) Набувачу майнові права на твір (назва твору, вид твору та інші відомості, що дають змогу ідентифікувати твір) *комп'ютерна програма «Застосунок продажу нерухомості»* (далі —Твір) на весь строк дії авторського права, на всю територію світу, а саме:
право на використання Твору;
виключне право дозволяти використання Твору;
право перешкоджати неправомірному використанню Твору, в тому числі забороняти таке використання (варіант 1, у випадку повної передачі прав).

Відчужувач частково передає (відчужує) Набувачу майнові права на твір (назва твору) *комп'ютерна програма «Застосунок продажу нерухомості»* (далі - Твір) на весь строк дії авторського права, на всю територію світу, а саме:

- право на використання Твору способами:

перекладу твору;

переробки твор;

(інші способи, передбачені статею 441 Цивільного кодексу України, статтею 15 Закону України „Про авторське право і суміжні права")

повне необмежене беззастережне використання твору у будь-який спосіб без жодних легальних наслідків.

- виключне право дозволяти використання Твору зазначеними способами;
- право перешкоджати неправомірному використанню Твору, в тому числі забороняти таке використання зазначеними способами (варіант 2, у випадку часткової передачі прав).

Набувач приймає такі майнові права та зобов'язується оплатити їх відповідно до умов цього Договору.

2.3. У результаті передання-прийняття майнових прав Відчужувач втрачає будь-які майнові права, а Набувач отримує право:

2.3.1. Використовувати Твір способами, передбаченими Цивільним кодексом України, Законом України „Про авторське право і суміжні права" (варіант 1, у випадку повної передачі прав).

Використовувати Твір способами, зазначеними у пункті 2.1 цього Договору (варіант 2, у випадку часткової передачі).

2.3.2. Дозволяти третім особам використовувати Твір способами, передбаченими Цивільним кодексом України, Законом України „Про авторське право і суміжні права"; перешкоджати неправомірному використанню Твору, в тому числі забороняти таке використання (варіант 1, у випадку повної передачі прав).

Дозволяти третім особам використовувати Твір способами, зазначеними у пункті 2.1 цього Договору; перешкоджати неправомірному використанню Твору, в тому числі забороняти таке використання зазначеними способами (варіант 2, у випадку часткової передачі прав).

2.3.3. Передавати (відчужувати) повністю або частково права на Твір третім особам (варіант 1, у випадку повної передачі прав).

Передавати (відчужувати) повністю або частково права на Твір третім особам способами, зазначеними у пункті 2.1 цього Договору (варіант 2, у випадку часткової передачі).

2.4. Після підписання цього Договору Відчужувач не має права передавати (відчужувати) майнові права на Твір, а також передавати права на використання Твору третім особам (варіант 1, у випадку повної передачі прав).

Після підписання цього Договору Відчужувач не має права передавати (відчужувати) майнові права на Твір, а також передавати права на використання Твору третім особам способами, зазначеними у пункті 2.1 цього Договору (варіант 2, у випадку часткової передачі).

3. ПЕРЕДАЧА ТВОРУ

3.1. Передача Твору Відчужувачем Набувачу здійснюється шляхом (зазначити, яким чином має передаватися Твір, на якому матеріальному носії, в електронному вигляді тощо) *в електронному та будь-якому іншому можливому відвореному вигляді* в день підписання цього Договору за Актом приймання-передачі Твору, який підписується Сторонами і є невід'ємною частиною цього Договору.

3.2. На момент передання Твору Набувачу Відчужувач гарантує, що:

лише йому належать виключні майнові права на Твір;

майнові права на Твір повністю або частково не передано (не відчужено) третім особам;

майнові права на Твір (повністю або частково) не є предметом застави, судового спору або претензій з боку третіх осіб.

4. ВИПЛАТА ВИНАГОРОДИ

4.1. За придбання майнових прав на Твір Набувач сплачує Відчужувачу винагороду в формі одноразового (паушального) платежу в розмірі *0* грн.

4.2. Виплата винагороди здійснюється у безготівковій (готівковій) формі, шляхом перерахування грошових коштів на розрахунковий рахунок Відчужувача (іншим чином), протягом *1* (не правда, як приклад) робочих днів з дня підписання сторонами Акта приймання-передачі Твору.

5. ВІДПОВІДАЛЬНІСТЬ СТОРІН ДОГОВОРУ

5.1. Сторона, яка не виконала або неналежним чином виконала зобов'язання за цим Договором, повинна відшкодувати іншій Стороні завдані збитки в повному обсязі.
5.2. У випадку передання (відчуження) майнових прав на Твір або передачі прав на використання Твору Відчужувачем третім особам, Відчужувач сплачує штраф у розмірі *0*% від суми, зазначеної у пункті 4.1 цього Договору.
5.3. У випадку порушення строку розрахунку, вказаного в пункті 4.2 цього Договору, Набувач сплачує Відчужувачу пеню в розмірі *0* за кожен день прострочення.

6. ВИРІШЕННЯ СПОРІВ

6.1. Сторони зобов'язуються вирішувати будь-який спір шляхом переговорів і в досудовому порядку.
6.2. У разі неможливості вирішення спору шляхом переговорів та в досудовому порядку, спір може бути передано для вирішення у судовому порядку.

7. ФОРС-МАЖОРНІ ОБСТАВИНИ

7.1. Сторони не несуть відповідальності за невиконання або неналежне виконання умов Договору за умов виникнення форс-мажорних обставин.
7.2. По завершенні таких обставин Сторона, яка не виконала свої зобов'язання за Договором, повинна виконати їх у терміни, на які було призупинено виконання зобов'язань.
7.3. Сторона не має права посилатися на форс-мажорні обставини у випадку, коли вони розпочалися у момент прострочення нею виконання її обов'язків за Договором.

8. ІНШІ УМОВИ

8.1. Договір набирає чинності з моменту його підписання Сторонами.
8.2. Положення Договору щодо виплати винагороди є конфіденційною інформацією, яку жодна з Сторін не має права розголошувати без попередньої згоди іншої Сторонни, крім випадків, передбачених законодавством.
8.3. Будь-які зміни і доповнення до цього Договору дійсні за умови, що вони вчинені у письмовій формі та підписані Сторонами.

8.4. З усіх питань, не передбачених цим Договором, Сторони керуються чинним законодавством України.

8.5. Договір укладений в двох примірниках, які мають однакову юридичну силу, по одному кожній із Сторін.

МІСЦЕЗНАХОДЖЕННЯ І РЕКВІЗИТИ СТОРІН:

Відчужувач: *Запоріжжя, Україна* Набувач: *громадськість по всьому світу*

ФРАГМЕНТ КОДУ ПРОГРАМИ

```
public class Database {
    string query;
    MySqlCommand command;
    MySqlDataReader reader;
    MySqlConnection connection;
    public const string LastCreatedID="SELECT LAST_INSERT_ID();";

    // SYSTEM
    public Database(MySqlConnection connection) { this.connection =
connection; }
    public MySqlDataReader read(string query){
        command=new MySqlCommand(query,connection);
        return command.ExecuteReader();
    }
    public void write(string query){
        command = new MySqlCommand(query, connection);
        command.ExecuteNonQuery();
    }

    // CREATE
    public User createUser(string name, int admin = 0, int balance=0) {
        write($"INSERT INTO user (name,admin,balance) VALUES
('{name}',{admin},{balance});");
        reader=read(LastCreatedID);
        var user = new User();
        while (reader.Read()) {
            user.ID = reader.GetInt32(0);
            user.Name = name;
            user.Admin = admin;
            user.Balance=balance;
        }
        reader.Close();
        return user;
    }
    public Estate createEstate(string title, string kind, User owner,
int price) {
        write($"INSERT INTO estate (title,kind,owner_id,price) VALUES
('{title}','{kind}',{owner.ID},{price});");
```

```
        reader=read(LastCreatedID);
        var estate = new Estate();
        while (reader.Read()) {
            estate.ID = reader.GetInt32(0);
            estate.Title = title;
            estate.Kind = kind;
            estate.Owner = owner;
            estate.Price=price;
        }
        reader.Close();
        return estate;
    }
    public Meeting createMeeting(User sender,Estate target){
        write($"INSERT INTO meeting (sender_id,target_id) VALUES
({sender.ID},{target.ID});");
        reader=read(LastCreatedID);
        var meeting = new Meeting();
        while (reader.Read()) {
            meeting.ID = reader.GetInt32(0);
            meeting.Sender=sender;
            meeting.Target=target;
        }
        reader.Close();
        return meeting;
    }

    // READ
    public List<User> getUsers(){
        reader=read("SELECT id,name,admin,balance FROM user;");
        var users=new List<User>();
        while (reader.Read()) {
            var user = new User();
            user.ID = reader.GetInt32(0);
            user.Name = reader.GetString(1);
            user.Admin = reader.GetInt32(2);
            user.Balance=reader.GetInt32(3);
            users.Add(user);
        }
        reader.Close();
        return users;
    }
    public User getUser(int id){
        return
getUsers().Where(u=>u.ID==id).ToList().ElementAtOrDefault(0);
    }
    public List<Estate> getEstates(){
        reader=read("SELECT id,owner_id,title,kind,price FROM
estate;");
        var estates=new List<Estate>();
        var owners=new List<int>();
        while (reader.Read()){
            var estate=new Estate();
            estate.ID=reader.GetInt32(0);
            owners.Add(reader.GetInt32(1));
            estate.Title=reader.GetString(2);
            estate.Kind=reader.GetString(3);
            estate.Price=reader.GetInt32(4);
            estates.Add(estate);
```

```
        }
        reader.Close();
        for (int i=0;i<estates.Count;i++){
            estates[i].Owner=getUser(owners[i]);
        }
        return estates;
    }
    public Estate getEstate(int id){
        return
getEstates().Where(e=>e.ID==id).ToList().ElementAtOrDefault(0);
    }
    public List<Meeting> getMeetings(){
        reader=read("SELECT id,sender_id,target_id,score,status FROM
meeting;");
        var meetings=new List<Meeting>();
        var senders = new List<int>();
        var targets = new List<int>();
        while (reader.Read()) {
            var meeting = new Meeting();
            meeting.ID = reader.GetInt32(0);
            senders.Add(reader.GetInt32(1));
            targets.Add(reader.GetInt32(2));
            meeting.Score = reader.GetString(3);
            meeting.Status = reader.GetString(4);
            meetings.Add(meeting);
        }
        reader.Close();
        for (int i = 0; i < meetings.Count; i++) {
            meetings[i].Sender = getUser(senders[i]);
            meetings[i].Target = getEstate(targets[i]);
        }
        return meetings;
    }
    public Meeting getMeeting(int id){
        return
getMeetings().Where(m=>m.ID==id).ToList().ElementAtOrDefault(0);
    }

    // UPDATE
    public void updateUser(User user) {
        write($"UPDATE user SET
name='{user.Name}',admin={user.Admin},balance={user.Balance} WHERE
id={user.ID};");
    }
    public void updateEstate(Estate estate) {
        write($"UPDATE estate SET
owner_id={estate.Owner.ID},title='{estate.Title}',kind='{estate.Kind}',
price={estate.Price} WHERE id={estate.ID};");
    }
    public void updateMeeting(Meeting meeting) {
        write($"UPDATE meeting SET
sender_id={meeting.Sender.ID},target_id={meeting.Target.ID},score='{mee
ting.Score}',status='{meeting.Status}' WHERE id={meeting.ID};");
    }

    // DELETE
    public void deleteUser(int id) {
        write($"DELETE FROM user WHERE id={id};");
```

```
    }
    public void deleteEstate(int id) {
        write($"DELETE FROM estate WHERE id={id};");
    }
    public void deleteMeeting(int id) {
        write($"DELETE FROM meeting WHERE id={id};");
    }
}
```

ПРЕЗЕНТАЦІЯ

Застосунок купівлі та продажу нерухомості

Практична робота з дисципліни “Розробка прикладних програм”

Студент КНТ-122: Онищенко О. А.
Викладач кафедри ПЗ: Зелік О. В.

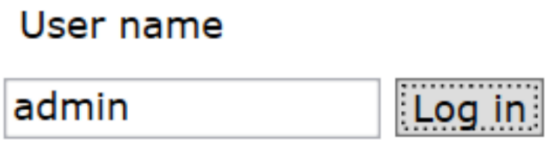
Цілі

- Продаж нерухомості
- Купівля нерухомості
- Призначення зустрічей на перегляд нерухомості

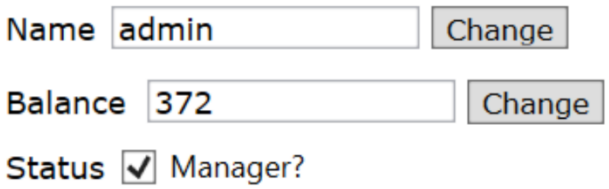
План

1. Сторінка входу за іменем користувача
2. Можливість редагувати дані профілю
3. Можливість перегляду та купівлі оголошень
4. Можливість призначення зустрічей
5. Можливість зміни оголошень
6. Можливість зміни зустрічей
7. Можливість продажу оголошень

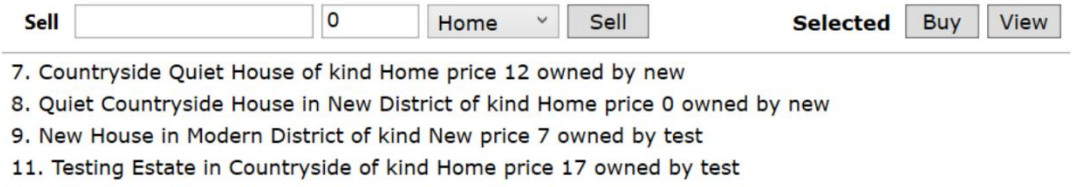
Вхід



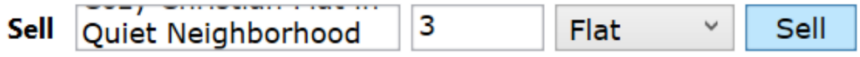
Профіль



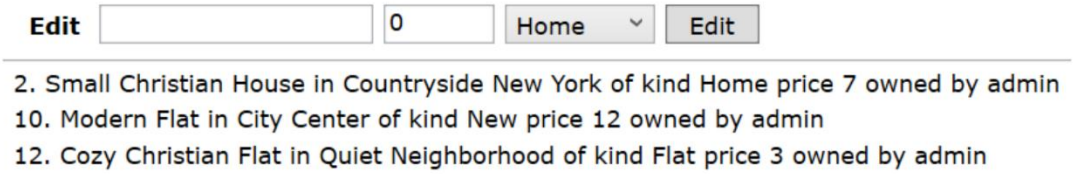
Ринок оголошень



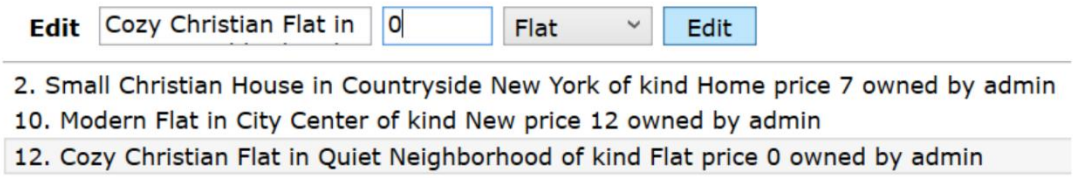
Додавання оголошення на ринку



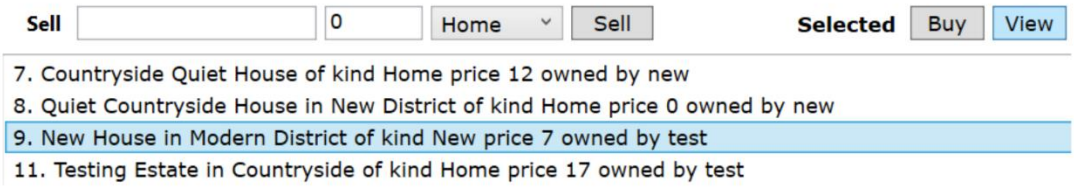
Перегляд наявних оголошень



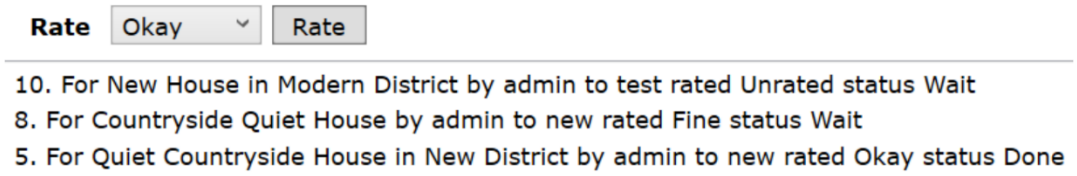
Зміна даних оголошення



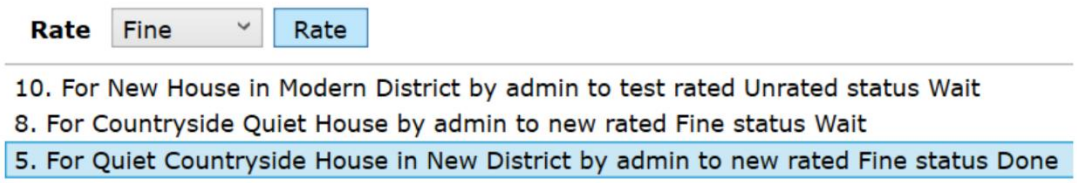
Призначення зустрічі на оголошення



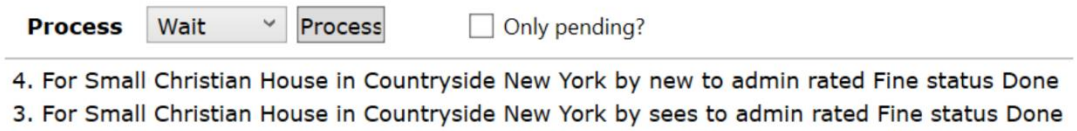
Перегляд призначених зустрічей



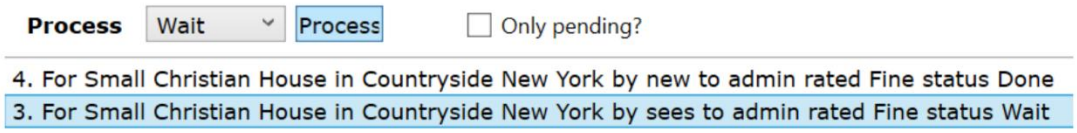
Зміна оцінки зустрічі



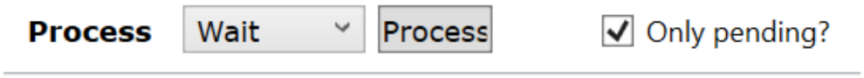
Перегляд вхідних зустрічей



Зміна статусу зустрічі



Перегляд лише очікуючих вхідних зустрічей



Так бо полюбив Бог сьвіт, що Сина свого єдинородного дав, щоб кожен, віруючий в Него, не погиб, а мав життє вічнє. Йоана 3:16

ДОДАТОК А – КОД ПРОГРАМИ

A1 – Database.cs

```
using System.Collections.Generic;
using MySql.Data.MySqlClient;
using System.Linq;

namespace seven {
    public class Database {
        string query;
```

```
        MySqlCommand command;
        MySqlDataReader reader;
        MySqlConnection connection;
        public const string LastCreatedID="SELECT LAST_INSERT_ID();";

        // SYSTEM
        public Database(MySqlConnection connection){ this.connection =
connection; }
        public MySqlDataReader read(string query){
            command=new MySqlCommand(query,connection);
            return command.ExecuteReader();
        }
        public void write(string query){
            command = new MySqlCommand(query, connection);
            command.ExecuteNonQuery();
        }

        // CREATE
        public User createUser(string name, int admin = 0, int
balance=0){
            write($"INSERT INTO user (name,admin,balance) VALUES
('{name}',{admin},{balance});");
            reader=read(LastCreatedID);
            var user = new User();
            while (reader.Read()){
                user.ID = reader.GetInt32(0);
                user.Name = name;
                user.Admin = admin;
                user.Balance=balance;
            }
            reader.Close();
            return user;
        }
        public Estate createEstate(string title, string kind, User
owner, int price){
            write($"INSERT INTO estate (title,kind,owner_id,price)
VALUES ('{title}','{kind}',{owner.ID},{price});");
            reader=read(LastCreatedID);
            var estate = new Estate();
            while (reader.Read()){
                estate.ID = reader.GetInt32(0);
                estate.Title = title;
                estate.Kind = kind;
                estate.Owner = owner;
                estate.Price=price;
            }
            reader.Close();
            return estate;
        }
        public Meeting createMeeting(User sender,Estate target){
            write($"INSERT INTO meeting (sender_id,target_id) VALUES
({sender.ID},{target.ID});");
            reader=read(LastCreatedID);
            var meeting = new Meeting();
            while (reader.Read()){
                meeting.ID = reader.GetInt32(0);
                meeting.Sender=sender;
                meeting.Target=target;
```

```
            }
            reader.Close();
            return meeting;
        }

        // READ
        public List<User> getUsers(){
            reader=read("SELECT id,name,admin,balance FROM user;");
            var users=new List<User>();
            while (reader.Read()){
                var user = new User();
                user.ID = reader.GetInt32(0);
                user.Name = reader.GetString(1);
                user.Admin = reader.GetInt32(2);
                user.Balance=reader.GetInt32(3);
                users.Add(user);
            }
            reader.Close();
            return users;
        }
        public User getUser(int id){
            return
getUsers().Where(u=>u.ID==id).ToList().ElementAtOrDefault(0);
        }
        public List<Estate> getEstates(){
            reader=read("SELECT id,owner_id,title,kind,price FROM
estate;");
            var estates=new List<Estate>();
            var owners=new List<int>();
            while (reader.Read()){
                var estate=new Estate();
                estate.ID=reader.GetInt32(0);
                owners.Add(reader.GetInt32(1));
                estate.Title=reader.GetString(2);
                estate.Kind=reader.GetString(3);
                estate.Price=reader.GetInt32(4);
                estates.Add(estate);
            }
            reader.Close();
            for (int i=0;i<estates.Count;i++){
                estates[i].Owner=getUser(owners[i]);
            }
            return estates;
        }
        public Estate getEstate(int id){
            return
getEstates().Where(e=>e.ID==id).ToList().ElementAtOrDefault(0);
        }
        public List<Meeting> getMeetings(){
            reader=read("SELECT id,sender_id,target_id,score,status
FROM meeting;");
            var meetings=new List<Meeting>();
            var senders = new List<int>();
            var targets = new List<int>();
            while (reader.Read()){
                var meeting = new Meeting();
                meeting.ID = reader.GetInt32(0);
                senders.Add(reader.GetInt32(1));
```

```
                targets.Add(reader.GetInt32(2));
                meeting.Score = reader.GetString(3);
                meeting.Status = reader.GetString(4);
                meetings.Add(meeting);
            }
            reader.Close();
            for (int i = 0; i < meetings.Count; i++){
                meetings[i].Sender = getUser(senders[i]);
                meetings[i].Target = getEstate(targets[i]);
            }
            return meetings;
        }
        public Meeting getMeeting(int id){
            return
getMeetings().Where(m=>m.ID==id).ToList().ElementAtOrDefault(0);
        }

        // UPDATE
        public void updateUser(User user){
            write($"UPDATE user SET
name='{user.Name}',admin={user.Admin},balance={user.Balance} WHERE
id={user.ID};");
        }
        public void updateEstate(Estate estate){
            write($"UPDATE estate SET
owner_id={estate.Owner.ID},title='{estate.Title}',kind='{estate.Kind}',
price={estate.Price} WHERE id={estate.ID};");
        }
        public void updateMeeting(Meeting meeting){
            write($"UPDATE meeting SET
sender_id={meeting.Sender.ID},target_id={meeting.Target.ID},score='{mee
ting.Score}',status='{meeting.Status}' WHERE id={meeting.ID};");
        }

        // DELETE
        public void deleteUser(int id){
            write($"DELETE FROM user WHERE id={id};");
        }
        public void deleteEstate(int id){
            write($"DELETE FROM estate WHERE id={id};");
        }
        public void deleteMeeting(int id){
            write($"DELETE FROM meeting WHERE id={id};");
        }
    }
}
```

A2 – Estate.cs

```
namespace seven {
    public class Estate {
        public int ID { get; set; }
        public User Owner { get; set; }
        public string Title { get; set; }
        public string Kind { get; set; }
```

```
        public int Price { get; set; } = 0;
    }
}
```

A3 – EstateKind.cs

```
namespace seven {
    public static class EstateKind {
        public const string Home = "Home";
        public const string Flat = "Flat";
        public const string New = "New";
    }
}
```

A4 – LoginPage.xaml

```
<Page x:Class="seven.LoginPage"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:seven"
      mc:Ignorable="d"
      Title="LoginPage" Height="85" Width="199">

    <Grid>
        <TextBox x:Name="UserNameInput" HorizontalAlignment="Left"
Margin="10,40,0,0" TextWrapping="Wrap" VerticalAlignment="Top"
Width="120" Height="19" FontFamily="Verdana"/>
        <Button Content="Log in" HorizontalAlignment="Left"
Margin="135,40,0,0" VerticalAlignment="Top" FontFamily="Verdana"
Width="47" Click="Button_Click"/>
        <Label Content="User name" HorizontalAlignment="Left"
Margin="10,10,0,0" VerticalAlignment="Top" FontFamily="Verdana"
Width="90" FontStretch="Normal"/>

    </Grid>
</Page>
```

A5 – LoginPage.xaml.cs

```
using System.Windows;
using System.Windows.Controls;

namespace seven {
    /// <summary>
    /// Interaction logic for LoginPage.xaml
```

```
    /// </summary>
    public partial class LoginPage : Page {
        public LoginPage(){
            InitializeComponent();
            UserNameInput.Focus();
        }
        private void Button_Click(object sender, RoutedEventArgs e){
            var userName = UserNameInput.Text;
            this.NavigationService.Navigate(new ProfilePage(userName));
        }
    }
}
```

A6 – Meeting.cs

```
namespace seven {
    public class Meeting {
        public int ID { get; set; }
        public User Sender { get; set; }
        public Estate Target { get; set; }
        public string Score { get; set; } = "Unrated";
        public string Status { get; set; } = MeetingStatus.Wait;
    }
}
```

A7 – MeetingScore.cs

```
namespace seven {
    public static class MeetingScore {
        public const string Bad = "Bad";
        public const string Okay = "Okay";
        public const string Fine = "Fine";
    }
}
```

A8 – MeetingStatus.cs

```
namespace seven {
    public static class MeetingStatus {
        public const string Wait = "Wait";
        public const string Done = "Done";
        public const string Skip = "Skip";
    }
}
```

A9 – ProfilePage.xaml

```
<Page x:Class="seven.ProfilePage"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:seven"
      mc:Ignorable="d"
      d:DesignHeight="450" d:DesignWidth="800"
      Title="ProfilePage">
    <TabControl>
        <TabItem Selector.Selected="homeTabOpened" x:Name="HomeTab"
Header="Home">
            <Grid Background="White">
                <Label x:Name="UserNameLabel" Content="Name"
HorizontalAlignment="Left" Margin="10,15,0,0" VerticalAlignment="Top"
FontFamily="Verdana"/>
                <Label x:Name="UserBalanceLabel" Content="Balance"
HorizontalAlignment="Left" Margin="10,44,0,0" VerticalAlignment="Top"
FontFamily="Verdana"/>
                <Button x:Name="ChangeNameButton" Content="Change"
HorizontalAlignment="Left" Margin="181,19,0,0" VerticalAlignment="Top"
Height="17" Width="53" FontFamily="Verdana" FontSize="11"
Click="ChangeNameButton_Click"/>
                <TextBox x:Name="UserNameBox"
HorizontalAlignment="Left" Margin="56,19,0,0" TextWrapping="Wrap"
VerticalAlignment="Top" Width="120" FontFamily="Verdana"/>
                <TextBox x:Name="UserBalanceBox"
HorizontalAlignment="Left" Margin="70,48,0,0" TextWrapping="Wrap"
VerticalAlignment="Top" Width="120" FontFamily="Verdana"/>
                <Button x:Name="ChangeBalanceButton" Content="Change"
HorizontalAlignment="Left" Margin="195,48,0,0" VerticalAlignment="Top"
Height="17" Width="53" FontFamily="Verdana" FontSize="11"
Click="ChangeBalanceButton_Click"/>
                <CheckBox x:Name="UserStatusToggle" Content="Manager?"
HorizontalAlignment="Left" Margin="59,74,0,0" VerticalAlignment="Top"
Click="UserStatusToggle_Click" />
                <Label x:Name="UserManagerLabel" Content="Status"
HorizontalAlignment="Left" Margin="10,69,0,0" VerticalAlignment="Top"
FontFamily="Verdana"/>
            </Grid>
        </TabItem>
        <TabItem Selector.Selected="availableEstatesTabOpened"
x:Name="AvailableEstatesTab" Header="Available Estate">
            <Grid Background="White">
                <ListBox x:Name="AvailableEstatesContainer"
d:ItemsSource="{d:SampleData ItemCount=5}" Margin="0,35,0,0"
FontFamily="Verdana"/>
                <Button x:Name="BuyEstateButton" Content="Buy"
HorizontalAlignment="Left" Margin="531,7,0,0" VerticalAlignment="Top"
FontFamily="Verdana" Width="43" Click="BuyEstateButton_Click"
Height="20"/>
                <Button x:Name="SetMeetingButton" Content="View"
HorizontalAlignment="Left" Margin="579,7,0,0" VerticalAlignment="Top"
```

```
FontFamily="Verdana" Width="43" Height="20"
Click="SetMeetingButton_Click"/>
                <TextBox x:Name="SellTitleInput"
HorizontalAlignment="Left" Margin="44,7,0,0" TextWrapping="Wrap"
VerticalAlignment="Top" Width="140" FontFamily="Verdana" Height="20"/>
                <TextBox x:Name="SellPriceInput"
HorizontalAlignment="Left" Margin="189,7,0,0" TextWrapping="Wrap"
Text="0" VerticalAlignment="Top" Width="57" FontFamily="Verdana"
Height="20"/>
                <ComboBox x:Name="SellKindInput"
HorizontalAlignment="Left" Margin="251,7,0,0" VerticalAlignment="Top"
Width="77" Height="20" FontFamily="Verdana">
                    <ComboBoxItem Content="Home" IsSelected="True"/>
                    <ComboBoxItem Content="Flat"/>
                    <ComboBoxItem Content="New"/>
                </ComboBox>
                <Button x:Name="SellButton" Content="Sell"
HorizontalAlignment="Left" Margin="333,7,0,0" VerticalAlignment="Top"
Height="20" FontFamily="Verdana" Width="48"
Click="AeSellButton_Click"/>
                <Label x:Name="AeSelectedHeading" Content="Selected"
HorizontalAlignment="Left" Margin="460,5,0,0" VerticalAlignment="Top"
FontFamily="Verdana" FontWeight="Bold"/>
                <Label x:Name="SellHeading" Content="Sell"
HorizontalAlignment="Left" Margin="10,4,0,0" VerticalAlignment="Top"
FontWeight="Bold"/>
            </Grid>
        </TabItem>
        <TabItem Selector.Selected="ownedEstatesTabOpened"
x:Name="OwnedEstatesTab" Header="Owned Estate">
            <Grid Background="White">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition/>
                </Grid.ColumnDefinitions>
                <ListBox x:Name="OwnedEstatesContainer"
d:ItemsSource="{d:SampleData ItemCount=5}" Margin="0,34,0,0"
FontFamily="Verdana"
SelectionChanged="OwnedEstatesContainer_SelectionChanged"/>
                <TextBox x:Name="EditTitleInput"
HorizontalAlignment="Left" Margin="51,6,0,0" TextWrapping="Wrap"
VerticalAlignment="Top" Width="140" FontFamily="Verdana" Height="20"/>
                <TextBox x:Name="EditPriceInput"
HorizontalAlignment="Left" Margin="196,6,0,0" TextWrapping="Wrap"
Text="0" VerticalAlignment="Top" Width="57" FontFamily="Verdana"
Height="20"/>
                <ComboBox x:Name="EditKindInput"
HorizontalAlignment="Left" Margin="258,6,0,0" VerticalAlignment="Top"
Width="77" Height="20" FontFamily="Verdana">
                    <ComboBoxItem Content="Home" IsSelected="True"/>
                    <ComboBoxItem Content="Flat"/>
                    <ComboBoxItem Content="New"/>
                </ComboBox>
                <Button x:Name="EditButton" Content="Edit"
HorizontalAlignment="Left" Margin="340,6,0,0" VerticalAlignment="Top"
Height="20" FontFamily="Verdana" Width="48"
Click="OeEditButton_Click"/>
```

```
                <Label x:Name="EditHeading" Content="Edit"
HorizontalAlignment="Left" Margin="10,4,0,0" VerticalAlignment="Top"
FontWeight="Bold" FontFamily="Verdana"/>
            </Grid>
        </TabItem>
        <TabItem Selector.Selected="incomingMeetingsTabOpened"
x:Name="IncomingMeetingsTab" Header="Incoming Meetings">
            <Grid Background="White">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition/>
                </Grid.ColumnDefinitions>
                <ListBox x:Name="IncomingMeetingsContainer"
d:ItemsSource="{d:SampleData ItemCount=5}" Margin="0,40,0,0"
FontFamily="Verdana"
SelectionChanged="IncomingMeetingsContainer_SelectionChanged"/>
                <Button x:Name="ProcessButton" Content="Process"
HorizontalAlignment="Left" Margin="162,12,0,0" VerticalAlignment="Top"
Height="20" FontFamily="Verdana" Width="48"
Click="ProcessButton_Click"/>
                <Label x:Name="ProcessHeading" Content="Process"
HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top"
FontWeight="Bold" FontFamily="Verdana"/>
                <ComboBox x:Name="ProcessInput"
HorizontalAlignment="Left" Margin="80,12,0,0" VerticalAlignment="Top"
Width="77" Height="20" FontFamily="Verdana">
                    <ComboBoxItem Content="Wait" IsSelected="True"
FontFamily="Verdana"/>
                    <ComboBoxItem Content="Done" FontFamily="Verdana"/>
                    <ComboBoxItem Content="Skip" FontFamily="Verdana"/>
                </ComboBox>
            </Grid>
        </TabItem>
        <TabItem Selector.Selected="outgoingMeetingsTabOpened"
x:Name="OutgoingMeetingsTab" Header="Outgoing Meetings">
            <Grid Background="White">
                <ListBox x:Name="OutgoingMeetingsContainer"
d:ItemsSource="{d:SampleData ItemCount=5}" Margin="0,38,0,0"
FontFamily="Verdana"
SelectionChanged="OutgoingMeetingsContainer_SelectionChanged"/>
                <Button x:Name="RateButton" Content="Rate"
HorizontalAlignment="Left" Margin="138,10,0,0" VerticalAlignment="Top"
Height="20" FontFamily="Verdana" Width="48" Click="RateButton_Click"/>
                <Label x:Name="RateHeading" Content="Rate"
HorizontalAlignment="Left" Margin="10,8,0,0" VerticalAlignment="Top"
FontWeight="Bold" FontFamily="Verdana"/>
                <ComboBox x:Name="RateInput" HorizontalAlignment="Left"
Margin="56,10,0,0" VerticalAlignment="Top" Width="77" Height="20"
FontFamily="Verdana">
                    <ComboBoxItem Content="Bad" FontFamily="Verdana"/>
                    <ComboBoxItem Content="Okay" FontFamily="Verdana"
IsSelected="True"/>
                    <ComboBoxItem Content="Fine" FontFamily="Verdana"/>
                </ComboBox>
            </Grid>
        </TabItem>
    </TabControl>
</Page>
```

A10 – ProfilePage.xaml.cs

```
using System;
using System.Linq;
using System.Windows;
using MySql.Data.MySqlClient;
using System.Windows.Controls;

namespace seven {
    /// <summary>
    /// Interaction logic for ProfilePage.xaml
    /// </summary>
    public partial class ProfilePage : Page
    {
        public User client;
        public MySqlConnection connection = new
MySqlConnection(UtilityVariables.connectionString);
        public Database database;
        public ProfilePage(string userName){
            InitializeComponent();
            connection.Open();
            database = new Database(connection);

            User found=database.getUsers().Where(u => u.Name ==
userName).ToList().ElementAtOrDefault(0);
            if (found != null){
                client = found;
            } else {
                client = database.createUser(userName);
            }
        }
        public void homeTabOpened(object sender, RoutedEventArgs e){
            showUserData();
        }
        public void ownedEstatesTabOpened(object sender,
RoutedEventArgs e){
            showOwnedEstates();
        }
        public void availableEstatesTabOpened(object sender,
RoutedEventArgs e){
            showAvailableEstates();
        }
        public void incomingMeetingsTabOpened(object sender,
RoutedEventArgs e){
            showIncomingMeetings();
        }
        public void outgoingMeetingsTabOpened(object sender,
RoutedEventArgs e){
            showOutgoingMeetings();
        }
        public void showUserData(){
            UserNameBox.Text=client.Name;
            UserBalanceBox.Text= client.Balance.ToString();
            UserStatusToggle.IsChecked=Convert.ToBoolean(client.Admin);
        }
        public void showOwnedEstates(){
```

```
            var data = database.getEstates().Where(e => e.Owner.ID ==
client.ID).ToList();
            OwnedEstatesContainer.Items.Clear();
            foreach (var e in data){
                OwnedEstatesContainer.Items.Add($"{e.ID}. {e.Title} of
kind {e.Kind} price {e.Price} owned by {e.Owner.Name}");
            }
            EditTitleInput.Text = "";
            EditKindInput.Text = EstateKind.Home;
            EditPriceInput.Text = "0";
        }
        public void showAvailableEstates(){
            var data =
database.getEstates().Where(e=>e.Owner.ID!=client.ID).ToList();
            AvailableEstatesContainer.Items.Clear();
            foreach (var e in data){
                AvailableEstatesContainer.Items.Add($"{e.ID}. {e.Title}
of kind {e.Kind} price {e.Price} owned by {e.Owner.Name}");
            }
        }
        public void showIncomingMeetings(bool onlyPending=false){
            var data = database.getMeetings().Where(m =>
m.Target.Owner.ID == client.ID).OrderBy(m=>m.ID).Reverse().ToList();
            if (onlyPending) { data = data.Where(m => m.Status ==
MeetingStatus.Wait && m.Score == "Unrated").ToList(); }
            IncomingMeetingsContainer.Items.Clear();
            foreach (var m in data){
                IncomingMeetingsContainer.Items.Add($"{m.ID}. For
{m.Target.Title} by {m.Sender.Name} to {m.Target.Owner.Name} rated
{m.Score} status {m.Status}");
            }
        }
        public void showOutgoingMeetings(){
            var data =
database.getMeetings().Where(m=>m.Sender.ID==client.ID).OrderBy(m =>
m.ID).Reverse().ToList();
            OutgoingMeetingsContainer.Items.Clear();
            foreach (var m in data){
                OutgoingMeetingsContainer.Items.Add($"{m.ID}. For
{m.Target.Title} by {m.Sender.Name} to {m.Target.Owner.Name} rated
{m.Score} status {m.Status}");
            }
        }
        private void ChangeNameButton_Click(object sender,
RoutedEventArgs e){
            var name = UserNameBox.Text;
            client.Name = name;
            database.updateUser(client);

            client = database.getUser(client.ID);
            showUserData();
        }
        private void ChangeBalanceButton_Click(object sender,
RoutedEventArgs e){
            int balance=client.Balance;
            try {
                balance = Convert.ToInt32(UserBalanceBox.Text);
```

```
            } catch { MessageBox.Show("Wrong balance. Please enter a
number"); }

            client.Balance = balance;
            database.updateUser(client);

            client = database.getUser(client.ID);
            showUserData();
        }
        private void UserStatusToggle_Click(object sender,
RoutedEventArgs e){
            var status = UserStatusToggle.IsChecked;
            client.Admin = Convert.ToInt32(status);
            database.updateUser(client);

            client = database.getUser(client.ID);
            showUserData();
        }
        private void BuyEstateButton_Click(object sender,
RoutedEventArgs e){
            if (AvailableEstatesContainer.SelectedIndex<0){ return; }
            string id =
AvailableEstatesContainer.SelectedValue.ToString().Split('.')[0];
            var estate = database.getEstate(int.Parse(id));

            if (client.Balance < estate.Price){
                MessageBox.Show("Not enough money");
                return;
            }
            estate.Owner = client;
            client.Balance -= estate.Price;

            database.updateEstate(estate);
            showAvailableEstates();
        }
        private void SetMeetingButton_Click(object sender,
RoutedEventArgs e){
            if (AvailableEstatesContainer.SelectedIndex < 0){ return; }
            string id =
AvailableEstatesContainer.SelectedValue.ToString().Split('.')[0];
            var estate = database.getEstate(int.Parse(id));

            database.createMeeting(client, estate);
            showOutgoingMeetings();
        }
        private void AeSellButton_Click(object sender, RoutedEventArgs
e){
            var title = SellTitleInput.Text;
            var priceInput = SellPriceInput.Text;
            int price;
            var kind = SellKindInput.Text;
            try {
                price = int.Parse(priceInput);
            } catch {
                MessageBox.Show("Incorrect price, please enter a
number");
                return;
            }
```

```
            if (kind==EstateKind.New && client.Admin == 0){
                MessageBox.Show($"Estate of kind {EstateKind.New} may
be added only by managers");
                return;
            }

            database.createEstate(title, kind, client, price);
            showAvailableEstates();
        }
        private void OwnedEstatesContainer_SelectionChanged(object
sender, SelectionChangedEventArgs e){
            if (OwnedEstatesContainer.SelectedIndex<0){ return; }
            string id =
OwnedEstatesContainer.SelectedValue.ToString().Split('.')[0];
            var estate = database.getEstate(int.Parse(id));

            EditTitleInput.Text = estate.Title;
            EditKindInput.Text = estate.Kind;
            EditPriceInput.Text = estate.Price.ToString();
        }
        private void OeEditButton_Click(object sender, RoutedEventArgs
e){
            if (OwnedEstatesContainer.SelectedIndex < 0){ return; }
            string id =
OwnedEstatesContainer.SelectedValue.ToString().Split('.')[0];
            var estate = database.getEstate(int.Parse(id));

            var title = EditTitleInput.Text;
            var kind = EditKindInput.Text;
            var priceInput = EditPriceInput.Text;
            int price;
            try {
                price = int.Parse(priceInput);
            } catch {
                MessageBox.Show("Incorrect price, please enter a
number");
                return;
            }
            estate.Title = title;
            estate.Price = price;
            estate.Kind = kind;

            database.updateEstate(estate);
            showOwnedEstates();
        }
        private void ProcessButton_Click(object sender, RoutedEventArgs
e){
            if (IncomingMeetingsContainer.SelectedIndex < 0){ return; }
            string id =
IncomingMeetingsContainer.SelectedValue.ToString().Split('.')[0];
            var meeting = database.getMeeting(int.Parse(id));

            var status = ProcessInput.Text;
            meeting.Status = status;

            database.updateMeeting(meeting);
            showIncomingMeetings();
        }
```

```
        private void RateButton_Click(object sender, RoutedEventArgs
e){
            if (OutgoingMeetingsContainer.SelectedIndex < 0){ return; }
            string id =
OutgoingMeetingsContainer.SelectedValue.ToString().Split('.')[0];
            var meeting = database.getMeeting(int.Parse(id));

            var score = RateInput.Text;
            meeting.Score = score;

            database.updateMeeting(meeting);
            showOutgoingMeetings();
        }
        private void IncomingMeetingsContainer_SelectionChanged(object
sender, SelectionChangedEventArgs e){
            if (IncomingMeetingsContainer.SelectedIndex < 0){ return; }
            string id =
IncomingMeetingsContainer.SelectedValue.ToString().Split('.')[0];
            var meeting = database.getMeeting(int.Parse(id));

            ProcessInput.Text = meeting.Status;
        }
        private void OutgoingMeetingsContainer_SelectionChanged(object
sender, SelectionChangedEventArgs e){
            if (OutgoingMeetingsContainer.SelectedIndex < 0){ return; }
            string id =
OutgoingMeetingsContainer.SelectedValue.ToString().Split('.')[0];
            var meeting = database.getMeeting(int.Parse(id));

            RateInput.Text = meeting.Score;
        }
        private void CheckBox_Click(object sender, RoutedEventArgs e) {
            var status =
Convert.ToBoolean(pendingMeetingsToggle.IsChecked);
            showIncomingMeetings(status);
        }
    }
}
```

A11 – User.cs

```
namespace seven {
    public class User {
        public int ID { get; set; }
        public string Name { get; set; }
        public int Admin { get; set; } = 0;
        public int Balance { get; set; } = 0;
    }
}
```

A12 – UtilityFunctions.cs

```
namespace seven {
    public class UtilityFunctions {
        public bool checkEstateKind(string kind){
            return kind != EstateKind.Home && kind != EstateKind.Flat
&& kind != EstateKind.New ? false : true;
        }
        public bool checkMeetingStatus(string status){
            return status != MeetingStatus.Wait && status !=
MeetingStatus.Done && status != MeetingStatus.Skip ? false : true;
        }
        public bool checkMeetingScore(string score){
            return score != MeetingScore.Bad && score !=
MeetingScore.Okay && score != MeetingScore.Fine ? false : true;
        }
    }
}
```

A13 – UtilityVariables.cs

```
namespace seven {
    public static class UtilityVariables {
        public const string connectionString =
"uid=root;pwd=1313;host=localhost;port=3306;database=fr_data";
    }
}
```