МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет
«Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт
з дисципліни

“Фреймворки розробки програмного забезпечення”

для студентів спеціальності
121 “Інженерія програмного забезпечення”
(денної форми навчання)

2023

Методичні вказівки до виконання лабораторних робіт з дисципліни “Фреймворки розробки програмного забезпечення” для студентів спеціальності 121 “Інженерія програмного забезпечення” (денної форми навчання) / В.М. Льовкін, О.О. Олійник, А.О. Олійник. – Запоріжжя : НУ «Запорізька політехніка», 2023. – 50 с.

Автори: В.М. Льовкін, к.т.н., доцент
О.О. Олійник, к.т.н., доцент
А.О. Олійник, д.т.н., професор

Рецензент: С.О. Субботін, д.т.н., професор

Відповідальний за випуск: С.О. Субботін, д.т.н., професор

Затверджено
на засіданні кафедри
програмних засобів

Протокол № 8
від “30” березня 2023 р.

ЗМІСТ

ВСТУП

Дане видання призначене для вивчення та практичного засвоєння студентами всіх форм навчання основ розробки програмного забезпечення на основі використання фреймворків.

Відповідно до графіка студенти перед виконанням лабораторної роботи повинні ознайомитися з конспектом лекцій та рекомендованою літературою. Дані методичні вказівки містять тільки основні, базові теоретичні відомості, необхідні для виконання лабораторних робіт, тому для виконання лабораторної роботи та при підготовці до її захисту необхідно ознайомитись з конспектом лекцій та опрацювати весь необхідний матеріал, наведений в переліку рекомендованої літератури, використовуючи також статті в інтернет-виданнях та актуальних наукових журналах з програмної інженерії.

Для одержання заліку з кожної роботи студент повинен у відповідності зі всіма наведеними вимогами розробити програмне забезпечення та оформити звіт, після чого продемонструвати на комп'ютері розроблене програмне забезпечення з виконанням всіх запропонованих викладачем тестів.

Звіт виконують на білому папері формату А4 (210 х 297 мм). Текст розміщують тільки з однієї сторони листа. Поля сторінки з усіх боків – 20 мм. Аркуші вміщують у канцелярський файл.

Усі завдання повинні виконуватись студентами індивідуально і не містити ознак плагіату як в оформленому звіті так і в розробленому програмному забезпеченні.

Під час співбесіди при захисті лабораторної роботи студент повинен виявити знання щодо мети роботи, теоретичного матеріалу, методів виконання кожного етапу роботи, змісту основних розділів звіту з демонстрацією результатів на конкретних прикладах, практичних прийомів використання теоретичного матеріалу в розробленому програмному забезпеченні. Студент повинен вміти обґрунтувати всі прийняті ним рішення та правильно аналізувати і використовувати на практиці отримані результати.

Для базової самоперевірки при підготовці до виконання і захисту роботи студент повинен відповісти на контрольні запитання, наведені наприкінці опису відповідної роботи.

1 ЛАБОРАТОРНА РОБОТА № 1 РОЗРОБЛЕННЯ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Мета роботи

Засвоїти основні принципи та засади оформлення технічного завдання на розробку програмного забезпечення.

1.2 Короткі теоретичні відомості

Технічне завдання (ТЗ) є основним документом усього проєкту та усіх взаємовідносин замовника і розробника. Коректне ТЗ, написане й погоджене усіма зацікавленими та відповідальними особами, є запорукою успішної реалізації проєкту. Після того, як проведено обговорення основних завдань проєкту, який має бути реалізовано, визначено їх основний перелік у відповідності з вимогами замовника, необхідно скласти й погодити технічне завдання.

ТЗ стандартно повинно містити такі розділи:

– вступ;

– підстави для розробки;

– призначення розробки;

– вимоги до програми чи програмному виробу;

– вимоги до програмної документації;

– техніко-економічні показники;

– стадії та етапи розробки;

– порядок контролю та приймання [24].

За необхідності в технічне завдання можна включати потрібні додатки.

У залежності від особливостей програми чи програмного виробу допускається уточнювати зміст розділів, вводити нові розділи чи поєднувати окремі з них [24].

У розділі «Вступ» вказують найменування, коротку характеристику області застосування програми чи програмного виробу та об’єкта, у якому використовують програму чи програмний виріб [24].

У розділі «Підстави для розробки» повинно бути зазначено:

– документ (документи), на підставі якого ведеться розроблення;

– організація, що затвердила цей документ, і дата його затвердження;

– найменування і (або) умовне позначення теми розробки [24].

У розділі «Призначення розробки» повинно бути зазначене функціональне та експлуатаційне призначення програми чи програмного виробу [24].

Розділ «Вимоги до програми чи програмного виробу» повинен містити наступні підрозділи:

– вимоги до функціональних характеристик;

– вимоги до надійності;

– умови експлуатації;

– вимоги до складу та параметрів технічних засобів;

– вимоги до інформаційної та програмної сумісності;

– вимоги до маркування та упакування;

– вимоги до транспортування та збереження;

– спеціальні вимоги [24].

У підрозділі «Вимоги до функціональних характеристик» повинно бути зазначено вимоги до складу виконуваних функцій, організації вхідних і вихідних даних (перелічені всі вхідні й вихідні дані та зазначено всі відомі вимоги до їх організації), тимчасових характеристик тощо, у даному підрозділі обов'язково має бути зазначено, якими групами користувачів та яким чином (з зазначенням відповідних функцій програми) планується використання програмного забезпечення [24].

У підрозділі «Вимоги до надійності» повинно бути зазначено вимоги до забезпечення надійного функціонування (забезпечення стійкого функціонування, контролю початкової та вихідної інформації, часу відновлення після відмовлення тощо), у даному підрозділі визначається, які саме збійні ситуації, ситуації невірного введення даних користувачем тощо мають оброблятися програмою [24].

У підрозділі «Умови експлуатації» повинно бути зазначено умови експлуатації (температура навколишнього повітря, відносна вологість тощо для обраних типів носіїв даних), при яких повинні забезпечуватися задані характеристики, вид обслуговування, необхідна кількість і кваліфікація персоналу (з зазначенням задач, які такий персонал повинен виконувати) [24].

У підрозділі «Вимоги до складу і параметрів технічних засобів» указують необхідний склад технічних засобів із зазначенням їхніх основних технічних характеристик [24].

У підрозділі «Вимоги до інформаційної і програмної сумісності» повинно бути зазначено вимоги до інформаційних структур на вході і виході та методів розв'язання, вихідних кодів, мов програмування та програмних засобів, що використовуються програмою: за необхідності повинен забезпечуватися захист інформації та програм [24].

У підрозділі «Вимоги до маркування та упакування» у загальному випадку указують вимоги до маркування програмного виробу, варіанти та способи упакування [24].

У підрозділі «Вимоги до транспортування та збереження» повинні бути зазначені для програмного виробу умови транспортування, місця збереження, умови збереження, умови складування, терміни збереження в різних умовах [24].

У підрозділі «Спеціальні вимоги» у даній лабораторній роботі потрібно навести вимоги до графічного інтерфейсу користувача [24]. Для цього потрібно виконати попереднє проєктування інтерфейсу. Для таких цілей існує багато доволі простих і зручних програмних засобів, частина з яких є безкоштовними. Наприклад, програма Pencil надає набір інструментів для проєктування інтерфейсів десктопних систем (рис. 1.1), мобільних додатків, вебдодатків.

У розділі «Вимоги до програмної документації» повинен бути зазначений попередній склад програмної документації і, за необхідності, спеціальні вимоги до неї [24].

У розділі «Техніко-економічні показники» повинні бути зазначені: орієнтована економічна ефективність, передбачувана річна потреба, економічні переваги розробки порівняно з кращими вітчизняними та закордонними аналогами [24].

У розділі «Стадії та етапи розробки» установлюють необхідні стадії розроблення, етапи та зміст робіт (перелік програмних документів, що повинні бути розроблені, погоджені та затверджені), а також, як правило, терміни розроблення, та визначають виконавців [24].

У розділі «Порядок контролю та приймання» повинні бути зазначені види іспитів і загальні вимоги до приймання роботи, у даному розділі можуть такою зазначатися конкретні випробування, які мають проводитися для контролю та приймання програмного продукту [24].

При підготовці матеріалів даного розділу використано джерело [24].

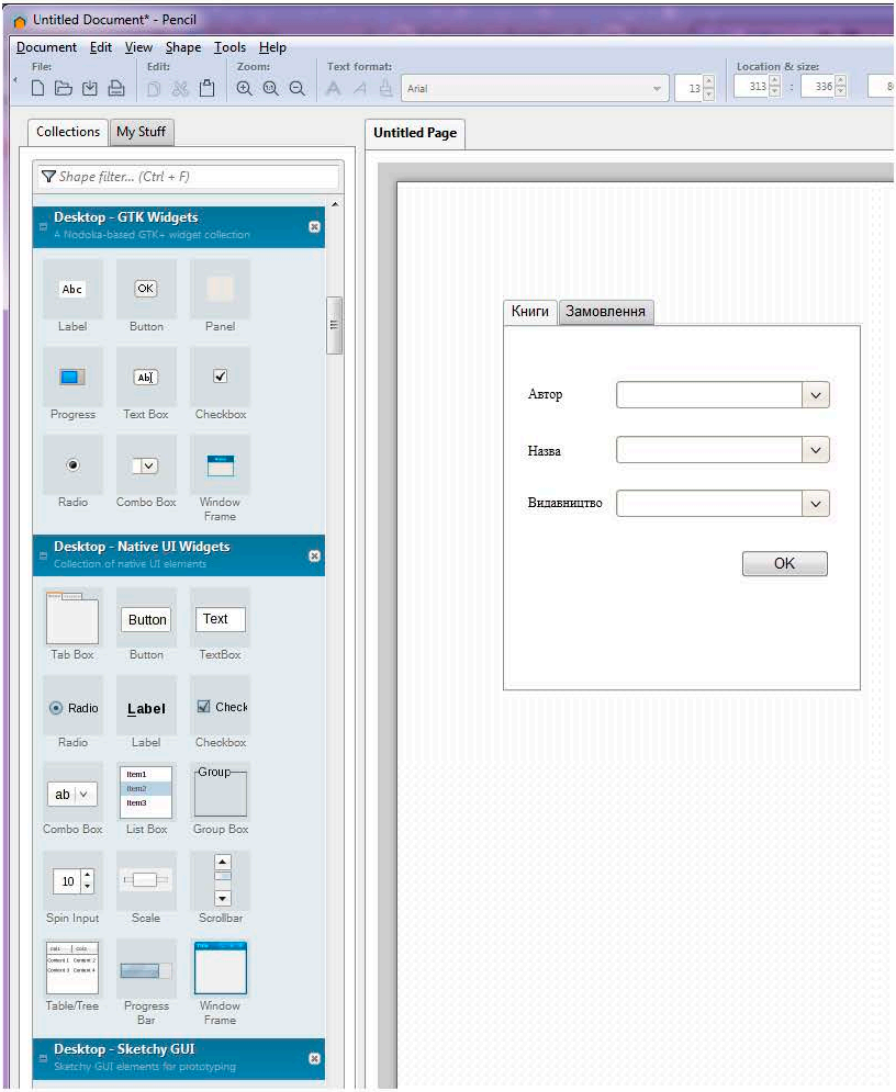

Рисунок 1.1 – Інструменти проєктування інтерфейсів у програмі Pencil

1.3 Завдання до роботи

1.3.1 Ознайомитися з літературою та основними теоретичними відомостями, необхідними для написання ТЗ на розробку програмного забезпечення.

1.3.2 Узгодити з викладачем індивідуальне завдання для розроблення програмного забезпечення з додатку А. Необхідно враховувати, що отримана тема та відповідні завдання мають ґрунтуватися на наступних принципах: розроблювана у подальшому система має працювати зі сховищем даних (базою даних), має виконувати обробку та збереження даних, повинна забезпечуватися можливість використання великою кількістю різних груп користувачів, а також повинна мати візуальний користувацький інтерфейс.

1.3.3 Розробити ТЗ у відповідності до отриманого індивідуального завдання, а також відповідно до всіх вимог, які висуваються до ТЗ, та викладених теоретичних відомостей.

1.3.4 Провести критичний аналіз розробленого ТЗ на предмет відображення всіх вимог до розроблюваного програмного забезпечення та єдиного підходу до їх формулювання, а також на предмет правильно встановлених часових меж розробки. За необхідності внести корективи.

1.3.5 Оформити звіт з роботи.

1.3.6 Відповісти на контрольні запитання.

1.4 Зміст звіту

1.4.1 Мета роботи.

1.4.2 Предметна область, мета розроблення програмного забезпечення та задачі, які має виконувати розроблюваний програмний продукт.

1.4.3 Розроблене технічне завдання відповідно до теоретичних відомостей, предметної області, мети та завдань, що ставляться перед системою.

1.4.4 Висновки, що містять відповіді на контрольні запитання, а також відображують результати виконання роботи та їх критичний аналіз.

1.5 Контрольні запитання

1.5.1 З якою метою розробляється ТЗ?

1.5.2 Яку інформацію відображають у ТЗ?

1.5.3 Які основні складові, з яких має складатися ТЗ?

1.5.4 Як впливає якість розробленого ТЗ на кінцевий програмний продукт?

1.5.5 Хто має розробляти ТЗ?

1.5.6 Які засоби використовують для розроблення ТЗ?

1.5.7 Яким вимогам має відповідати ТЗ?

1.5.8 Якими особливостями характеризуються ТЗ у відповідності до особливостей розроблюваного програмного проєкту?

1.5.9 Яку інформацію надає ТЗ виконавцям та замовнику?

1.5.10 Які стандарти визначають склад та вимоги до ТЗ?

2 ЛАБОРАТОРНА РОБОТА № 2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ

2.1 Мета роботи

Вивчити базові принципи та основи дослідження предметної області, у межах якої розробляється програмне забезпечення, і навчитися виконувати проєктування архітектури системи на базі проведеного дослідження з застосуванням шаблону проєктування MVVM.

2.2 Короткі теоретичні відомості

Архітектура – це високорівнева частина проєкту програмного додатку, каркас, що складається з деталей проєкту. Для опису архітектури проєкту достатньо стандартним рішенням є специфікація архітектури.

При визначенні архітектури програмного забезпечення має використовуватися шаблон проєктування Model-View-ViewModel (MVVM), що є одним з найбільш широко використовуваних шаблонів проєктування на даний момент.

У шаблоні MVVM є три основні компоненти:

– модель: класи моделей – це невізуальні класи, які інкапсулюють дані програми, відповідно модель можна розглядати як представлення моделі домену програми, яка зазвичай включає модель даних разом із бізнес-логікою та логікою перевірки;

– представлення відповідає за визначення структури, компонування та зовнішнього вигляду того, що користувач бачить на екрані, в ідеальному випадку кожне представлення даних визначається мовою XAML з обмеженим кодом, що не містить бізнес-логіки;

– модель представлення реалізує властивості та команди, до яких подання може прив'язувати дані, і сповіщає представлення про будь-які зміни стану через події сповіщень про зміни: властивості та команди, які надає модель представлення, визначають функціональні можливості, пропоновані користувацьким інтерфейсом, але представлення визначає, як ці функції мають відображатися.

Переваги використання шаблону MVVM:

– якщо існує реалізація моделі, яка інкапсулює існуючу бізнес-логіку, змінити її може бути важко або ризиковано, у такому сценарії модель представлення діє як адаптер для класів моделі та дає змогу уникнути будь-яких серйозних змін у коді моделі;

– розробники можуть створювати модульні тести для моделі представлення та моделі без використання представлення, модульні тести для моделі представлення можуть виконувати ті самі функції, що й у представленні;

– інтерфейс програми можна змінити, не торкаючись коду, за умови, що подання повністю реалізовано в XAML, тому нова версія представлення повинна працювати з існуючою моделлю представлення;

– дизайнери та розробники можуть працювати незалежно та одночасно над своїми компонентами в процесі розробки, дизайнери можуть зосередитися на представленні, а розробники можуть працювати над моделлю представлення та компонентами моделі.

Специфікація архітектури зокрема повинна містити наступну інформацію:

– альтернативні варіанти побудови архітектури, які було розглянуто, оцінено і за якими було прийнято рішення використовувати саме обраний підхід, принципи побудови архітектури;

– основні компоненти програми, що у випадку використання шаблону MVVM початково структурно визначаються його складовими, але також деталізуються аж до відповідних класів, які мають реалізовувати функціональні і інші вимоги до програми: кожна функція, яка була визначена в ТЗ, повинна бути чітко визначена серед компонентів, зокрема як окремий клас, як метод класу, сукупність методів або класів або як інша відповідна складова, в результаті кожна складова (компонент, підсистема, клас, метод) повинні бути чітко визначеними, а логіка їх відповідальності повинна бути окремо зазначена;

– правила і принципи взаємодії між складовими програми, які були визначені у попередньому пункті: сюди відносяться як правила організації цієї взаємодії, так і будь-які правила, що стосуються деякої заборони на взаємодію між складовими програми, якщо така заборона має місце (може зокрема визначатися шаблоном проєктування);

– формат роботи з даними, основні вимоги до цих даних, а також (за наявності) інформацію про наявні на поточний момент дані, принципи взаємодії з цими даними, шляхи подальшої їх інтеграції;

– питання організації безпеки мають бути визначені окремо, при цьому вирішення проблеми може бути виконано не тільки на основі створення відповідного власного програмного коду, але і шляхом вибору відповідних архітектурних рішень, рішень з вибору інструментів проєкту, які самі по собі вирішують певні потенційні проблеми, пов'язані з безпекою, які виникали б у випадку розроблення програмного забезпечення на основі інших підходів;

– результати аналізу і прийняті в підсумку рішення, що стосуються готовності системи до розширення в майбутньому, при цьому розширення може охоплювати розширення функціональних можливостей, кількості користувачів, обсягів даних;

– вимоги до продуктивності програмного забезпечення, його відмовостійкості в підсумку мають бути встановлені і проаналізовані для того, щоб їх можливо було забезпечити в підсумку;

– принципи і процедури повторного використання існуючих рішень під час реалізації програми.

Під час аналізу архітектури розроблюваного програмного проєкту потрібно дати відповіді на наступні запитання:

– чи ясно описана загальна організація програми? Чи включає специфікація грамотний огляд архітектури та її обґрунтування?

– чи адекватно визначено основні компоненти програми, їх області відповідальності й взаємодія з іншими компонентами?

– чи наведено опис найважливіших класів і їх обґрунтування?

– чи наведено опис організації даних і її обґрунтування?

– чи наведено опис організації й змісту сховища даних?

– чи визначені всі важливі бізнес-правила? Чи описано їх вплив на систему?

– чи описана стратегія проєктування GUI?

– чи зроблено GUI модульним, щоб його зміни не впливали на іншу частину програми?

– чи наведено опис стратегії введення-виведення даних та її обґрунтування?

– чи вказано оцінки ступеню використання обмежених ресурсів, таких як потоки, з'єднання зі сховищем даних, дескриптори, пропуск-

на спроможність мережі? Чи наведено опис стратегії керування такими ресурсами і її обґрунтування?

– чи описані вимоги до захищеності архітектури?

– чи визначає архітектура вимоги до обсягу й швидкодії всіх класів, підсистем і функціональних областей?

– чи описує архітектура спосіб досягнення масштабованості системи?

– чи розглянуті питання взаємодії системи з іншими системами?

– чи описана стратегія інтернаціоналізації/локалізації?

– чи визначена погоджена стратегія обробки помилок?

– чи визначений підхід до відмовостійкості системи (якщо це потрібно)?

– чи підтверджена можливість технічної реалізації всіх частин системи?

– чи визначений підхід до реалізації надлишкової функціональності?

– чи прийняті необхідні рішення відносно «придбання або створення» компонентів системи?

– чи описано у специфікації, як повторно використовуваний код буде адаптований до інших аспектів архітектури?

– чи зможе архітектура адаптуватися до ймовірних змін?

При підготовці матеріалів даного розділу використано джерела [15], [19], [20], [22].

2.3 Завдання до роботи

2.3.1 Ознайомитися з літературою та основними теоретичними відомостями, необхідними для виконання роботи.

2.3.2 Провести аналіз предметної області, у рамках якої буде працювати розроблюване програмне забезпечення: виділити основні об’єкти, що зустрічаються у предметній області; виділити основні компоненти майбутньої системи; проаналізувати функції, які виконують виділені об’єкти предметної області.

2.3.3 На основі проведеного аналізу провести проєктування архітектури розроблюваної системи з обов’язковим використанням шаблону проєктування **Model-View-ViewModel (MVVM)**. В результаті проведеного проєктування має бути отриманий робочий варіант архітектури системи, що має включати: пакети, модулі, класи, бібліотеки

тощо з описом призначення та функцій кожного з них у відповідності з принципами MVVM та принципами створення графічного інтерфейсу користувача на основі фреймворку WPF.

2.3.4 Виходячи з отриманої архітектури та описаних у ТЗ вимог до системи, виконати обґрунтований вибір засобів подальшої розробки.

2.3.5 Виконати критичний аналіз отриманих результатів. За необхідності внести корективи у розроблену архітектуру або обрані засоби.

2.3.6 Оформити звіт з роботи.

2.3.7 Відповісти на контрольні запитання.

2.4 Зміст звіту

2.4.1 Мета роботи.

2.4.2 Архітектура розроблюваного програмного проєкту у відповідності до предметної області та шаблону проєктування MVVM.

2.4.3 Аналіз розробленої архітектури. В аналізі мають бути надані відповіді на запитання, представлені в теоретичних відомостях.

2.4.4 Обґрунтований вибір засобів розробки.

2.4.5 Висновки, що містять відповіді на контрольні запитання, а також відображують результати виконання роботи та їх критичний аналіз.

2.5 Контрольні запитання

2.5.1 Що таке архітектура програмного продукту?

2.5.2 Яке місце займає проєктування та розробка архітектури програмного забезпечення у життєвому циклі програмного забезпечення?

2.5.3 Які основні складові архітектури?

2.5.4 Яким вимогам має відповідати архітектура програмного забезпечення?

2.5.5 Як визначається якість архітектури програмного забезпечення?

2.5.6 Скільки часу (у відношенні до загального часу розробки) має розроблятися архітектура програмного продукту?

2.5.7 Яким чином можна описати архітектуру?

2.5.8 Які існують архітектурні шаблони?

2.5.9 Які існують основні принципи проєктування архітектури?

2.5.10 Які існують архітектурні стилі?

2.5.11 Що таке шаблон проєктування MVVM і які складові входять до нього?

2.5.12 Чим відрізняється шаблон проєктування MVVM від інших існуючих шаблонів проєктування?

3 ЛАБОРАТОРНА РОБОТА № 3 РЕАЛІЗАЦІЯ ЗБЕРІГАННЯ, ВИДОБУВАННЯ ТА ОБРОБКИ ДАНИХ

3.1 Мета роботи

Ознайомитися з сучасними системами керування базами даних, можливостями створення з'єднань з базами даних через програмні додатки, мовою XML та навчитися на практиці використовувати бази даних в якості сховищ даних.

3.2 Короткі теоретичні відомості

Microsoft SQL Server є системою керування реляційними базами даних, що була розроблена компанією Microsoft. Важливою характеристикою системи є її здатність працювати віддалено, дозволяючи таким чином доступ до роботи з даними ширшій кількості потенційних або фактичних користувачів.

Зазвичай одночасно з самою системою керування базами даних використовується окремий інструмент Management Studio, чому сприяє те, що програма має графічний інтерфейс користувача для створення баз даних і об'єктів у базах даних, має редактор запитів для взаємодії з базами даних шляхом написання операторів Transact-SQL.

XML (Extensible Markup Language) – розширювана мова розмітки. Вона може використовуватися для зручної роботи з даними, коли ці дані можуть як сприйматися самою програмою, так і достатньо просто сприйматися без додаткових маніпуляцій з даними самим користувачем.

У мові C# простір імен System.Xml надає підтримку оброблення XML, що ґрунтується на стандартах. Відповідно у програму необхідно додати наступний оператор:

```
using System.Xml;
```

Приклад читання вмісту елементів та атрибутів:

```
StringBuilder output = new StringBuilder()
```

```
String xmlString =
            @"<bookstore>
        <book genre='fantasy' publication='2020-06-10' ISBN='9-
331271-19-2'>
          </book>
       </bookstore>";

    using (XmlReader read_XML = XmlReader.Create(new
StringReader(xmlString)))
    { read_XML.ReadToFollowing("book");
      read_XML.MoveToFirstAttribute();
      string resultG = reader.Value;
      output.AppendLine("Жанр книги: " + resultG);
      read_XML.ReadToFollowing("title");
      output.AppendLine("Назва книги: " +
read_XML.ReadElementContentAsString());
    }
```

При підготовці матеріалів даного розділу використано джерела [5], [6], [9], [11], [16], [17], [20], [22], [23].

3.3 Завдання до роботи

3.3.1 Ознайомитися з літературою та основними теоретичними відомостями, необхідними для виконання роботи.

3.3.2 Виконати аналіз ТЗ та розробленої архітектури системи щодо вимог до організації даних.

3.3.3 Прийняти рішення про систему збереження даних та обґрунтувати її.

3.3.4 Розробити структуру бази даних, необхідної для збереження даних системи. Додатково (не є обов'язковою вимогою) для зберігання деяких даних можуть використовуватися XML-файли або вони можуть бути створені як паралельне сховище.

3.3.5 Реалізувати функціональність програмного забезпечення, пов'язану зі взаємодією з даними. Під час реалізації взаємодії має обов'язково враховуватися можливість роботи з даними з віддалених пристроїв, що мінімально має підтримуватися на рівні відповідної системи керування базами даних.

3.3.6 Виконати тестування розробленого програмного забезпечення. У процесі тестування має обов'язково застосовуватись модульне тестування.

Тестування має виконуватися шляхом взаємодії з різними файлами (в яких зберігаються дані) визначеної структури на пристроях з різними апаратними характеристиками під керуванням різних операційних систем або версій операційних систем.

3.3.7 Виконати аналіз отриманих результатів тестування. У процесі аналізу отриманих результатів має бути порівняно результати, отримані під керування різних операційних систем (або їх версій) та на різних пристроях.

3.3.8 Оформити звіт з роботи.

3.3.9 Відповісти на контрольні запитання.

3.4 Зміст звіту

3.4.1 Мета роботи.

3.4.2 Структура бази даних.

3.4.3 Текст програми (з коментарями), що реалізує взаємодію зі сховищами даних.

3.4.4 Результати тестування розробленого програмного забезпечення з наведенням фрагментів відповідних сховищ даних.

3.4.5 Аналіз отриманих результатів тестування.

3.4.6 Висновки, що відображають особисто отримані результати виконання роботи, їх критичний аналіз та обґрунтування прийнятих рішень про систему збереження даних.

3.5 Контрольні запитання

3.5.1 Що таке XML?

3.5.2 Для чого використовується мова XML?

3.5.3 Яким чином організована структура XML-файлів?

3.5.4 Який елемент документу XML називається кореневим?

3.5.5 Які існують кодування документів XML?

3.5.6 Які існують способи читання XML-файлів та чим вони відрізняються?

3.5.7 У чому полягає модель DOM?

3.5.8 У чому полягає модель SAX?

3.5.9 Що таке парсер?

3.5.10 Яким чином відбувається програмна взаємодія з XML-файлами для обраної Вами мови програмування?

3.5.11 Яким чином виконується додавання, зміна та вилучення даних з бази даних через програмний додаток?

3.5.12 Що необхідно виконати для встановлення з'єднання з базою даних через програмний додаток?

3.5.13 Які існують сучасні системи керування базами даних і чим вони відрізняються?

3.5.14 Яким чином організована робота з даними в MVVM?

4 ЛАБОРАТОРНА РОБОТА № 4 РОЗРОБЛЕННЯ БАЗОВИХ МОДУЛІВ ПРОГРАМНОЇ СИСТЕМИ

4.1 Мета роботи

Навчитися аналізувати технічне завдання, предметну область, на основі проведеного аналізу й дослідження виявляти першочергові для реалізації функції програмного забезпечення та реалізовувати їх.

4.2 Короткі теоретичні відомості

На даному етапі роботи над програмою потрібно визначити спершу перелік базових функцій програми. При цьому слід зважати на те, що в основі побудови архітектури використовується шаблон проєктування MVVM, що призводить до того, що під час реалізації буде застосовуватися тільки та частина розробки, що охоплює моделі і моделі представлень додатку.

Окрім того обов'язково потрібно враховувати вимоги до якісних програм, які також безпосередньо впливають на роботу і на даному етапі:

– необхідно досягти чіткого розуміння принципів організації компонентів у складі шаблону проєктування MVVM і з'ясувати, які саме переваги надає безпосередньо під час написання коду використання даного шаблону, визначити бібліотеки, застосування яких для реалізації шаблону, спростить процес розробки, а тільки після чого починати реалізовувати основні функції програми: застосування шаблону проєктування на практиці не повинно призвести до надмірності в реалізації, коли виконується зайва робота, а створені рішення є непридатними для повторного використання, для підтримки модифікації системи через свою неоптимальність;

– варто враховувати принципи, описані в попередньому пункті, не тільки для застосування шаблону проєктування, але і для всіх інструментів, які були обрані для застосування в проєкті, включаючи мову програмування, засоби роботи з даними тощо;

– програма повинна дозволити досягти вимоги, які були початково встановлені для розробки, а реалізація всіх цих вимог повинна

бути направленою на реалізацію мети: оптимальність досягнення мети, реалізація запланованих функцій без виникнення помилок, збійних ситуацій, коректна обробка нестандартних ситуацій є важливими складовими досягнення мети;

– необхідно провести рефакторинг коду і його оптимізацію (якщо це необхідно за вимогами до програми) після того, як було отримано працюючу версію програми, що вже дозволяє вирішити поставлені перед нею завдання, таким чином сприяючи збільшенню життєвого циклу цієї програми в подальшому;

– якісне тестування програми, в тому числі використання модульних тестів, дозволяє зменшити ризики подальшого використання програми на практиці;

– якщо ресурси, виділені на проєкт, дозволяють, то потрібно передбачити особливі можливості роботи з програмою для користувачів, які використовують повільні, застарілі пристрої тощо, що має розширити потенційну базу користувачів програми, які при цьому отримають позитивний досвід роботи з нею.

При підготовці матеріалів даного розділу використано джерела [1]-[3], [7], [12]-[13], [15], [21].

4.3 Завдання до роботи

4.3.1 Ознайомитися з літературою та основними теоретичними відомостями, необхідними для виконання роботи.

4.3.2 Виконати аналіз ТЗ і розробленої архітектури системи та виділити першочергові функції, які має виконувати система.

4.3.3 Розробити програму, що реалізує першочергові функції, які має виконувати система. На даному етапі потрібно розробити консольний додаток (або декілька консольних додатків за необхідності). Реалізація програми виконується за допомогою обраних на попередньому етапі засобів розробки. Реалізована програма має відповідати нормам програмування, тобто має бути виключена надлишковість програмного коду тощо, що має забезпечуватися за допомогою використання принципів об’єктно-орієнтованого програмування: поліморфізм, інкапсуляція, спадкування.

4.3.4 Виконати тестування розробленого програмного забезпечення. У процесі тестування має обов’язково застосовуватись модульне тестування.

Тестування має виконуватися за різноманітних умов роботи програми, тобто шляхом введення різних даних, шляхом виконання на пристроях з різними апаратними характеристиками, а також під керуванням різних операційних систем або версій операційних систем.

Результатами тестування є швидкість роботи програми, вимоги до ресурсів, а також коректність отримуваних результатів.

Обов'язково мають бути створені модульні тести для перевірки результатів роботи за передбаченими ТЗ функціями програми.

4.3.5 Виконати аналіз отриманих результатів тестування. У процесі аналізу отриманих результатів має бути порівняно результати, отримані під керування різних операційних систем (або їх версій) та на різних пристроях.

4.3.6 Оформити звіт з роботи.

4.3.7 Відповісти на контрольні запитання.

4.4 Зміст звіту

4.4.1 Мета роботи.

4.4.2 Перелік першочергових функцій, які має виконувати система, та перелік усіх інших, передбачених для реалізації, функцій з обґрунтуванням вибору.

4.4.3 Текст програми (з коментарями), що реалізує базові функції розроблюваного програмного продукту.

4.4.4 Результати тестування розробленого програмного забезпечення. Результати тестування мають відображатися у графічній та табличній формі для різних умов запуску.

4.4.5 Аналіз отриманих результатів тестування.

4.4.6 Висновки, що відображають особисто отримані результати виконання роботи та їх критичний аналіз. У висновках має бути відображено необхідність реалізації обраних базових функцій програмного продукту.

4.5 Контрольні запитання

4.5.1 Яким чином необхідно виконувати аналіз предметної області?

4.5.2 З яких міркувань слід виділяти основні, базові функції програмного забезпечення?

4.5.3 Як необхідно будувати ієрархію класів для можливості розширення у майбутньому?

4.5.4 Назвіть дії, що необхідно виконувати для майбутньої передачі коду.

4.5.5 Який код можна вважати якісним?

4.5.6 Яких принципів необхідно дотримуватися в процесі реалізації якісного програмного забезпечення?

4.5.7 Яким чином необхідно виконувати тестування для забезпечення якості програмного продукту?

4.5.8 У яких випадках необхідно використовувати готові рішення в процесі розроблення програмного забезпечення?

4.5.9 У чому полягає модульне тестування?

4.5.10 Яким чином використання об'єктно-орієнтованої парадигми програмування може впливати на якість програмного забезпечення?

5 ЛАБОРАТОРНА РОБОТА № 5 РОЗШИРЕННЯ ФУНКЦІОНАЛЬНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ РОБОТИ В КОМАНДІ

5.1 Мета роботи

Навчитися розширювати функціональність вже існуючого програмного забезпечення, розробленого іншими розробниками.

5.2 Короткі теоретичні відомості

Розроблення нової функціональності для існуючого програмного забезпечення є доволі складним завданням, оскільки завжди є відмінність між початковими рішеннями та поточною реалізацією, а також між підходами до програмування різних програмістів або команд.

Для визначення поточного стану розробки додатка існують різні архітектурні інструменти, одними з яких є архітектурні інструменти в Visual Studio Team System 2010 або аналогічні засоби Visual Studio 2013, 2015, які дозволяють зрозуміти наявний додаток та прийняті в ньому рішення, спроєктувати необхідну нову функціональність і перевірити відповідність реалізації проєктуванню.

Визначення залежностей між частинами додатка може бути дуже важливим для пошуку потенційних проблем. До того ж, маючи візуальне представлення, можна знайти найкраще місце для впровадження нової функціональності в архітектурі системи.

У Visual Studio Team System можна одержати візуальне представлення рішень щодо збірки, просторів імен, класів або за налаштовуваним фільтром у вигляді документа в форматі Directed Graph Markup Language (DGML), використовуючи інструмент Generate Dependency Graph з пункту меню Architecture та його відповідні підрозділи By Assembly, By Namespace, By Class, Custom…

Створений на підставі рішення DGML-документ може бути представлений у декількох варіантах: Dependency Matrix, Force Directed Layout або Top to Bottom Layout. Кожний варіант пропонує своє власне представлення структури проєкту. Це високорівневе представ-

лення, яке надає можливість оцінити загальне представлення архітектури додатка.

Візуалізація залежностей може бути дуже корисною під час аналізу. Наприклад, можливість візуалізувати взаємовідношення класу з іншими класами дозволяє швидко визначити наявні засоби взаємодії між класами і набагато швидше відповідно внести зміни в потрібний клас. Надається також можливість візуалізації послідовності викликів, які відбуваються в ключових частинах додатка. Функція Generate Sequence Diagram, доступна через редактор коду, надає можливість спостерігати за викликами методів, які виконує додаток.

Аналогічні інструменти представлені в Visual Studio 2015, але всі вони в даному випадку об'єднані під назвою мапи коду (Code Map).

Для швидкого розуміння архітектури проєкту можна обрати з меню Architecture пункт Generate Code Map for Solution Without Building, що дозволить побудувати мапу, яка відображає граф структури рішення та посилання проєкту, що дозволяє зрозуміти взаємозв'язки між різними частинами рішення. На архітектурній мапі відображаються зв'язки, які зокрема демонструють наявність наслідування між класами в проєктах (зелений колір ліній). Фільтри, розташовані справа, дозволяють виключити деякі залежності: наприклад, приховати код, який знаходиться на стадії тестування. Можна збільшувати ділянки діаграми для подальшого дослідження або обирати вузол (вузли) для дослідження на іншій діаграмі, використовуючи пункт контекстного меню New Graph from Selection. Для кожного програмного компоненту, що відображається на мапі, можна деталізувати простори імен, класи, розглядаючи залежності всередині кожного структурного блоку.

Після того, як отримано більш-менш повне представлення про те, як працює існуючий додаток, можна більш ефективно розробити нову функціональність програмного забезпечення.

Діаграми Unified Modeling Language (UML) дозволяють представити структуру додатка в зрозумілому для інших вигляді. Наприклад, можна побудувати діаграми компонентів або діаграми класів UML, які описують існуючі елементи структури додатку, а потім додати в діаграми нові елементи, щоб проілюструвати й задокументувати зміни.

Діаграму класів (Class diagram) використовують для подання статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування.

Для створення діаграми класів у Visual Studio 2015 необхідно обрати з меню Architecture пункт New UML or Layer Diagram…, що дозволить створити нову діаграму, обираючи відповідний тип (у даному випадку UML Class Diagram). Для додавання на діаграму існуючих типів необхідно з браузера рішення обрати відповідний клас та з контекстного меню обрати пункт View → View Class Diagram.

Якщо на створеній діаграмі на будь-якому класі натиснути правою кнопкою, то до нього можна додати новий метод, властивість, поле, подію тощо, що призведе до автоматичного створення відповідного коду.

Діаграма компонентів описує особливості фізичного представлення системи. Дана діаграма застосовується для моделювання статичного вигляду системи з точки зору реалізації.

Діаграми Use Case застосовуються для моделювання вигляду системи з точки зору прецедентів (або варіантів використання). Даний вид діаграм використовується для моделювання динамічних аспектів системи.

При підготовці матеріалів даного розділу використано джерела [17], [19].

5.3 Завдання до роботи

5.3.1 Ознайомитися з літературою та основними теоретичними відомостями, необхідними для виконання роботи.

5.3.2 Узгодити з викладачем варіант індивідуального завдання, виконуваний іншим студентом, для виконання даної лабораторної роботи.

5.3.3 Одержати вихідні коди розробленого програмного забезпечення та розроблене технічне завдання.

5.3.4 Виконати аналіз архітектури системи, реалізованих функціональних характеристик та відповідність їх ТЗ. Виділити функції, що реалізовані у системі, та функції, ще не реалізовані у системі.

5.3.5 Реалізувати функції, що передбачені ТЗ та ще не реалізовані в отриманій програмній системі. При реалізації функцій необхід-

но дотримуватися принципу ідентичності, щоб нові функції принципово не відрізнялися від раніше розроблених.

5.3.6 Виконати тестування розробленого програмного забезпечення. У процесі тестування має обов'язково застосовуватись модульне тестування, направлене окремо на функції, розроблені власноруч, та на функції, розроблені попереднім розробником.

Тестування має виконуватися за різноманітних умов роботи програми, тобто шляхом введення різних даних, шляхом виконання на пристроях з різними апаратними характеристиками, а також під керуванням різних операційних систем або версій операційних систем.

Результатами тестування є швидкість роботи програми, вимоги до ресурсів, а також коректність отримуваних результатів.

5.3.7 Виконати аналіз отриманих результатів тестування. У процесі аналізу отриманих результатів має бути порівняно результати, отримані під керування різних операційних систем (або їх версій) та на різних пристроях.

5.3.8 Висунути пропозиції щодо розширення функціональності програмного забезпечення, орієнтуючись на розширення інноваційних характеристик програмного продукту.

5.3.9 Оформити звіт з роботи.

5.3.10 Відповісти на контрольні запитання.

5.4 Зміст звіту

5.4.1 Мета роботи.

5.4.2 Аналіз отриманого ТЗ та програмного забезпечення з обов'язковим наведенням діаграм і описом висновків.

5.4.3 Опис прийнятих рішень щодо розширення функціональності програмного забезпечення з використанням відповідних діаграм.

5.4.4 Текст програми (з коментарями), що реалізує розширення функціональності програмного продукту.

5.4.5 Результати тестування розробленого програмного продукту. Результати тестування мають відображатися у графічній та табличній формі для різних умов запуску.

5.4.6 Аналіз результатів тестування отриманої частини програмного забезпечення та власної розробки.

5.4.7 Пропозиції щодо розширення функціональності програмного забезпечення

5.4.8 Висновки, що відображають особисто отримані результати виконання роботи та їх критичний аналіз. У висновках має бути відображено функції, що вже були розроблено, та якість їх розроблення, необхідність розроблення нових функцій та прийняті щодо цього рішення, а також шляхи подальшого розвитку програмного проєкту.

5.5 Контрольні запитання

5.5.1 Яким чином необхідно проводити аналіз ТЗ?

5.5.2 Яким вимогам мають відповідати назви змінних, класів тощо для адекватного розуміння змісту відповідних елементів іншими розробниками?

5.5.3 Як слід виконувати аналіз існуючої програмної розробки?

5.5.4 Які Ви знаєте засоби, що дозволяють полегшити процес аналізу існуючого програмного забезпечення?

5.5.5 Що таке графи залежностей та мапи коду?

5.5.6 Які файли мають формат DGML?

5.5.7 Яким чином побудувати діаграму класів та яким чином її можна застосувати під час розширення функціональності програмного додатку?

5.5.8 Яким чином побудувати діаграму компонентів та для чого її можна використати під час розробки?

5.5.9 Яким чином побудувати діаграму послідовностей і яка інформація на ній наводиться?

5.5.10 Для чого призначені діаграми Use Case та яким чином їх побудувати в існуючому програмному проєкті?

6 ЛАБОРАТОРНА РОБОТА № 6 РЕАЛІЗАЦІЯ ІННОВАЦІЙНИХ ФУНКЦІЙ СИСТЕМИ

6.1 Мета роботи

Навчитися виділяти інноваційні функції системи, виконувати їх аналіз та реалізовувати на практиці.

6.2 Короткі теоретичні відомості

У сучасному світі практично в кожній області існує цілий ряд програмних продуктів, які вирішують деяку проблему. Окрім того програми, що вирішують різні проблеми, можуть бути практично функціонально схожими між собою, використовуючи однакові принципи побудови логіки роботи з функціями в цій предметній області. У таких випадках надважливою є наявність функцій, які можна назвати інноваційними, тобто таких, які будуть відрізняти програму від її конкурентів. У цьому сенсі багато сучасних програм, які досягли успіху в користувачів, можна вважати інноваційними, адже вони змогли надати саме такі засоби, які визначили перевагу цього продукту порівняно з іншими існуючими.

Для кожної інноваційної функції зокрема і інноваційного проєкту загалом потрібно визначити шлях впровадження, адже важливою є не тільки сама ідея, а найголовніше те, які ця ідея у вигляді відповідної функції буде на практиці застосовуватися кінцевими користувачами програми. До цих питань належать питання пошуку аудиторії, впровадження програми загалом і відповідної функції зокрема в виробництві, а не тільки в фактичній реалізації.

При підготовці матеріалів даного розділу використано джерело [18].

6.3 Завдання до роботи

6.3.1 Ознайомитися з літературою та основними теоретичними відомостями, необхідними для виконання роботи.

6.3.2 Одержати пропозиції щодо розширення інноваційної функціональності програмного забезпечення та виконати аналіз отриманих пропозицій.

6.3.3 Одержати вихідні коди розробленого програмного забезпечення.

6.3.4 Реалізувати інноваційні функції для розробленого програмного забезпечення.

6.3.5 Виконати тестування розробленого програмного забезпечення. У процесі тестування має обов'язково застосовуватись модульне тестування.

Тестування має виконуватися за різноманітних умов роботи програми, тобто шляхом введення різних даних, шляхом виконання на пристроях з різними апаратними характеристиками, а також під керуванням різних операційних систем або версій операційних систем.

6.3.6 Оформити звіт з роботи.

6.3.7 Відповісти на контрольні запитання.

6.4 Зміст звіту

6.4.1 Мета роботи.

6.4.2 Текст програми (з коментарями).

6.4.3 Результати тестування розробленого програмного забезпечення. Результати тестування мають відображатися у графічній та табличній формі для різних умов запуску.

6.4.4 Аналіз отриманих результатів тестування.

6.4.5 Висновки, що відображають особисто отримані результати виконання роботи та їх критичний аналіз. У висновках має бути відображено інноваційні функції, які було реалізовано, та виконано аналіз даного процесу.

6.5 Контрольні запитання

6.5.1 Який проєкт називають інноваційним?

6.5.2 З чого складається менеджмент інновацій?

6.5.3 Яким чином реалізація інноваційних функцій програмного продукту впливає на остаточний результат?

6.5.4 Яким чином інновації пов'язані з ризиками?

6.5.5 Що розуміють під стартапом?

6.5.6 Які розрізняють механізми росту програмних продуктів та в чому вони полягають?

6.5.7 Які розрізняють віражі розвитку програмного продукту та в чому вони полягають?

7 ЛАБОРАТОРНА РОБОТА № 7 РОЗРОБЛЕННЯ ЛЮДИННО-МАШИННОГО ІНТЕРФЕЙСУ ПРОГРАМНОЇ СИСТЕМИ

7.1 Мета роботи

Узагальнити та поглибити на практиці знання та навички розроблення графічного інтерфейсу програмної системи на основі використання фреймворку WPF.

7.2 Короткі теоретичні відомості

Windows Presentation Foundation (WPF) – фреймворк розробки інтерфейсу користувача, який використовує векторний механізм візуалізації, створений для використання переваг сучасного графічного обладнання, надає повний набір функцій розробки додатків, які включають мову Extensible Application Markup Language (XAML), елементи керування, зв'язування даних, макетування, 2D і 3D графіку, анімацію, стилі, шаблони, документи, медіа, текст.

Для реалізації безпосередньо зовнішнього вигляду програми використовується мова XAML, що дозволяє відділити зовнішній вигляд програми від всіх інших її складових (бізнес-логіки), що в результаті має такі переваги:

– витрати на розробку та обслуговування зменшуються, оскільки специфічна для зовнішнього вигляду розмітка не тісно пов'язана з кодом, що залежить від поведінки;

– розробка ефективніша, оскільки дизайнери можуть реалізовувати зовнішній вигляд програми одночасно з розробниками, які реалізують поведінку програми;

– глобалізація та локалізація для програм WPF спрощена.

Загалом XAML – це мова розмітки на основі XML, яка декларативно реалізує зовнішній вигляд програми. За допомогою цієї мови створюються вікна, діалогові вікна, сторінки і елементи керування користувача.

Стандартною основою побудови інтерфейсу є вікно:

<Window

```
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    Title="Вікно з кнопкою"
    Width="400" Height="300">
</Window>
```

Всередині вікна розміщується відповідні елементи керування. Наприклад, це може бути кнопка:

```
<Button Name="button">Кнопка</Button>
```

До вбудованих елементів керування WPF належать:

– кнопки: Button і RepeatButton;

– відображення даних: DataGrid, ListView та TreeView;

– відображення та вибір дати: Calendar та DatePicker;

– діалогові вікна: OpenFileDialog, PrintDialog і SaveFileDialog;

– цифрове чорнило: InkCanvas і InkPresenter;

– робота з документами: DocumentViewer, FlowDocumentPageViewer, FlowDocumentReader, FlowDocumentScrollViewer і StickyNoteControl;

– інструменти введення: TextBox, RichTextBox і PasswordBox;

– робота з макетом: Border, BulletDecorator, Canvas, DockPanel, Expander, Grid, GridView, GridSplitter, GroupBox, Panel, ResizeGrip, Separator, ScrollBar, ScrollViewer, StackPanel, Thumb, Viewbox, VirtualizingStackPanel, Window і WrapPanel;

– медіа: Image, MediaElement і SoundPlayerAction;

– меню: ContextMenu, Menu і ToolBar;

– навігація: Frame, Hyperlink, Page, NavigationWindow і TabControl;

– вибір: CheckBox, ComboBox, ListBox, RadioButton і Slider.

– інформація про користувача: AccessText, Label, Popup, ProgressBar, StatusBar, TextBlock і ToolTip.

При підготовці матеріалів даного підрозділу використано джерела [1], [2], [4], [7], [10] (зокрема [https://docs.microsoft.com/en-us/dotnet/desktop/wpf/overview/?view=netdesktop-6.0]).

7.3 Завдання до роботи

7.3.1 Ознайомитися з літературою та основними теоретичними відомостями, необхідними для виконання роботи.

7.3.2 Провести аналіз ТЗ на предмет вимог щодо візуального інтерфейсу програмного забезпечення.

7.3.3 Розробити візуальний інтерфейс у відповідності до вимог у ТЗ та попереднього проєктування інтерфейсу в ТЗ з обов'язковим використанням фреймворку створення інтерфейсу користувача WPF.

7.3.4 Виконати аналіз розробленого інтерфейсу відповідно до відомих вимог до інтерфейсів. Обґрунтувати прийняті рішення, базуючись на відомих правилах, законах, принципах, характеристиках.

7.3.4 Виконати тестування розробленого програмного забезпечення. Тестування має виконуватися на непідготовленому користувачі за різноманітних умов роботи програми, тобто шляхом введення різних даних, шляхом виконання на пристроях з різними апаратними характеристиками, а також під керуванням різних операційних систем або версій операційних систем та з різними параметрами налаштування монітору. Результатами тестування є швидкість роботи програми, вимоги до ресурсів, а також коректність отримуваних результатів та адекватність відображення інтерфейсу програми і взаємодії з користувачем.

7.3.5 Виконати аналіз отриманих результатів тестування. У процесі аналізу отриманих результатів має бути порівняно результати, отримані за різних умов виконання програми.

7.3.6 Оформити звіт з роботи.

7.3.7 Відповісти на контрольні запитання.

7.4 Зміст звіту

7.4.1 Мета роботи.

7.4.2 Аналіз ТЗ та розробленого програмного забезпечення щодо вимог до користувацького інтерфейсу з наведенням попередньо спроєктованого в ТЗ варіанту.

7.4.3 Результати розроблення користувацького інтерфейсу програмного забезпечення у вигляді відповідних форм.

7.4.4 Обґрунтування прийнятих рішень щодо інтерфейсу програмного забезпечення.

7.4.5 Текст програми (із коментарями), що реалізує користувацький інтерфейс.

7.4.6 Результати тестування розробленого програмного продукту. Результати тестування мають відображатися у графічній та табличній формі для різних умов запуску.

7.4.7 Аналіз отриманих результатів тестування.

7.4.8 Висновки, що відображають особисто отримані результати виконання роботи та їх критичний аналіз.

7.5 Контрольні запитання

7.5.1 Що розуміють під терміном юзабіліті?

7.5.2 Що розуміють під терміном дизайн?

7.5.3 Що розуміють під терміном інтерфейс та які існують види інтерфейсу?

7.5.4 Яке значення займають юзабіліті та дизайн при розробці користувацького інтерфейсу програмного забезпечення?

7.5.5 Виділіть рекомендації, у відповідності до яких слід розробляти дизайн користувацького інтерфейсу.

7.5.6 Назвіть основні правила юзабіліті.

7.5.7 Назвіть основні засоби проєктування користувацького інтерфейсу.

7.5.8 У чому полягає правило Фіттса?

7.5.9 Яким чином застосовується принцип Парето під час проєктування інтерфейсу?

7.5.10 Яким чином визначити взаємне розташування елементів інтерфейсу?

7.5.11 Які існують підходи до створення графічного інтерфейсу користувача?

7.5.12 В чому особливості фреймворку WPF?

7.5.13 Які переваги фреймворку WPF?

7.5.14 Які теги існують в мові XAML?

8 ЛАБОРАТОРНА РОБОТА № 8 ОФОРМЛЕННЯ АВТОРСЬКОГО СВІДОЦТВА НА РОЗРОБЛЕНУ ПРОГРАМУ

8.1 Мета роботи

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

8.2 Короткі теоретичні відомості

Відповідно до статті 18 Закону України «Про авторське право і суміжні права» комп'ютерні програми є набором інструкцій у вигляді слів, цифр, кодів, схем, символів чи в будь-якому іншому вигляді, виражених у формі, придатній для зчитування комп'ютером (настільним комп'ютером, ноутбуком, смартфоном, ігровою приставкою, смарт-телевізором тощо), які приводять його у дію для досягнення певної мети або результату, зокрема операційна система, прикладна програма, виражені у вихідному або об'єктному кодах, при цьому відносно охорони комп'ютерної програми визначено наступне:

– охорона комп'ютерної програми поширюється на комп'ютерні програми, виражені у вихідному або об'єктному кодах, якщо вони є оригінальними;

– охорона надається формі вираження комп'ютерної програми: графічний інтерфейс користувача, набір виконуваних функцій, формат файлів даних, які використовуються у комп'ютерній програмі для експлуатації її функцій, не є формами вираження комп'ютерної програми;

– ідеї та принципи, на яких ґрунтується будь-який елемент комп'ютерної програми, зокрема ті, на яких ґрунтується її інтерфейс, логічні схеми, алгоритми та мови програмування, не охороняються авторським правом [18].

Відносно права на бази даних (компіляції даних) визначено, що:

– бази даних (компіляції даних) охороняються авторським правом, якщо вони за добором та/або упорядкуванням їх складових частин є результатом творчої діяльності;

– охорона баз даних не поширюється і не завдає шкоди будь-яким правам на їх складові частини, що увійшли до складу баз даних;

– охорона баз даних не поширюється на комп'ютерні програми, що використовувалися під час створення або необхідні для функціонування баз даних;

– виробнику бази даних, крім бази, створеної для систематизації даних, що є публічною інформацією, який зробив якісно та/або кількісно значний внесок в отримання, перевірку чи подання вмісту бази даних, для запобігання вилученню та/або повторному використанню - будь-якому наданню невизначеному колу осіб повного вмісту бази даних або значної її частини у якісному чи кількісному вимірі, на таку базу даних надається право особливого роду;

– право особливого роду здійснюється незалежно від того, чи підлягає відповідна база даних охороні авторським правом або іншими правами: крім того, право особливого роду здійснюється незалежно від того, чи підлягає вміст відповідної бази даних охороні авторським правом або іншими правами, право особливого роду може передаватися (відчужуватися) відповідно до законодавства;

– охорона баз даних відповідно до права особливого роду не перешкоджає здійсненню прав, які існують щодо вмісту такої бази даних;

– право особливого роду починає діяти з дати закінчення створення бази даних та спливає через 15 років, що відліковуються з 1 січня року, наступного за роком створення бази даних;

– будь-яка істотна зміна вмісту бази даних у кількісному чи якісному вимірі, зокрема будь-яка істотна зміна, що випливає з накопичення доповнень, вилучень або послідовних змін, яка дозволяє зробити висновок, що йдеться про суттєвий, визначений якісно і кількісно внесок, надає базі даних, яка є результатом таких змін, окремий строк правової охорони [18].

При підготовці матеріалів даного підрозділу використано джерело [18].

8.3 Завдання до роботи

8.3.1 Ознайомитися з літературою та основними теоретичними відомостями, необхідними для виконання роботи.

8.3.2 Розглянути розроблену програмну систему на предмет відповідності поставленому ТЗ.

8.3.3 Оформити настанову щодо використання створеного програмного продукту, що має складатися з наступних елементів:

– призначення програми;
– структура системи;
– функціонування системи;
– структура та організація даних;
– основні функції;
– інтерфейсні засоби;
– вимоги до програмного та апаратного забезпечення;
– методика роботи користувача з системою.

8.3.4 Оформити заяву про реєстрацію авторського права на твір.

8.3.5 Оформити договір між роботодавцем та авторами комп'ютерної програми.

8.3.6 Створити презентацію для представлення результатів розробленого проєкту.

8.3.7 Оформити звіт з роботи.

8.3.8 Відповісти на контрольні запитання.

8.4 Зміст звіту

8.4.1 Мета роботи.

8.4.2 Аналіз ТЗ та розробленого програмного забезпечення.

8.4.3 Настанова щодо використання розробленої програмної системи.

8.4.4 Заява та договір.

8.4.5 Фрагмент тексту розробленої програмної системи.

8.4.6 Презентація програмного продукту.

8.4.7 Висновки, що відображають особисто отримані результати виконання роботи та їх критичний аналіз.

8.5 Контрольні запитання

8.5.1 Які законодавчі акти регламентують оформлення авторського свідоцтва на твір?

8.5.2 Хто має право на оформлення авторського свідоцтва на програму?

8.5.3 Чим є з законодавчої точки зору комп'ютерна програма?

8.5.4 Чим є з законодавчої точки зору база даних?

8.5.5 Які документи необхідно оформити для реєстрації авторського права на програму?

8.5.6 Хто займається оформленням документів під час реєстрації авторського права на програму?

8.5.7 Опишіть порядок реєстрації авторського свідоцтва на комп'ютерну програму.

8.5.8 Які органи державної влади приймають участь у процесі реєстрації та оформлення авторського права на програму?

8.5.9 У які строки приймається рішення щодо надання авторського свідоцтва на комп'ютерну програму?

8.5.10 Що представляє собою настанова щодо використання комп'ютерної програми?

ЛІТЕРАТУРА

1. Stonis, M. Enterprise Application Patterns using .NET MAUI [Текст] / M. Stonis. – Washington : One Microsoft Way, 2022. – 101 p.

2. Yuen, Sh. Mastering Windows Presentation Foundation [Текст] : Second Edition / Sh. Yuen. – Birmingham : Packt Publishing, 2020. – 636 p.

3. Model-View-ViewModel [Електронний ресурс]. – Режим доступу : https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm.

4. WPF Tutorial [Електронний ресурс]. – Режим доступу : https://www.tutorialspoint.com/wpf/index.htm.

5. Gorman, B. L. Practical Entity Framework [Текст] : Database Access for Enterprise Applications / Brian L. Gorman. – Berkeley : Apress, 2020. – 649 p.

6. Entity Framework Notes for Professionals [Електронний ресурс]. – 2018. – 94 с. – Режим доступу : https://goalkicker.com/EntityFrameworkBook/.

7. Weil, A. Learn WPF MVVM – XAML, C# and the MVVM pattern [Текст] : Be ready for coding away next week using WPF and MVVM / A. Weil. – Lulu.com, 2016. – 174 p.

8. Laurent, A. Understanding Open Source and Free Software Licensing [Текст] / Andrew M. St. Laurent. – O'Reilly Media, 2004. – 224 p.

9. Getting Started with WPF – EF Core [Електронний ресурс]. – Режим доступу : https://learn.microsoft.com/en-us/ef/core/get-started/wpf

10. What is Windows Presentation Foundation [Електронний ресурс]. – Режим доступу : https://learn.microsoft.com/en-us/dotnet/desktop/wpf/overview/?view=netdesktop-7.0.

11. Entity Framework documentation [Електронний ресурс]. – Режим доступу : https://learn.microsoft.com/en-us/ef/.

12. Prism Library [Електронний ресурс]. – Режим доступу : https://prismlibrary.com/.

13. Introduction to the MVVM Toolkit [Електронний ресурс]. – Режим доступу : https://learn.microsoft.com/en-us/dotnet/communitytoolkit/mvvm/.

14. Licenses – Open Source Initiative [Електронний ресурс]. – Режим доступу : https://opensource.org/licenses/.

15. The Model-View-ViewModel Pattern [Електронний ресурс]. – Режим доступу : https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm.

16. SQL Server technical documentation [Електронний ресурс]. – Режим доступу : https://docs.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver16.

17. Introducing Microsoft SQL Server 2019 [Текст] : Reliability, scalability, and security both on premises and in the cloud / Kellyn Gorman, Allan Hirt, Dave Noderer, Mitchell Pearson et al. – Birmingham : Packt Publishing, 2020. – 488 p.

18. Про авторське право і суміжні права [Електронний ресурс] : Закон України в редакції від 15 квітня 2023 року. – Режим доступу : https://zakon.rada.gov.ua/laws/show/2811-20#n855.

19. Analyze and model your architecture [Електронний ресурс]. – Режим доступу : https://msdn.microsoft.com/ru-ru/library/57b85fsc.aspx.

20. Kumar, V. MVVM (Model View ViewModel) Introduction [Електронний ресурс] : Part 1 / V. Kumar. – Режим доступу : https://www.c-sharpcorner.com/UploadFile/0b73e1/mvvm-model-view-viewmodel-introduction-part-1/.

21. Price, M. J. C# 9 and .NET 5 – Modern Cross-Platform Development [Текст] : Build intelligent apps, websites, and services with Blazor, ASP.NET Core, and Entity Framework Core using Visual Studio Code / M. J. Price. – Birmingham : Packt Publishing, 2020. – 822 p.

22. Johnson, B. Essential Visual Studio 2019 [Текст] : Boosting Development Productivity with Containers, Git, and Azure Tools / B. Johnson. – Apress, 2020. – 376 p.

23. Beginning XML [Текст] / David Hunter, Jeff Rafter, Joe Fawcet etc. ; 4 edition. – Wrox. – 2007. – 1080.

24. Комп'ютерні науки, інформаційні технології та інженерія програмного забезпечення [Текст] : навчальний посібник / під заг. ред. С.О. Субботіна. – Т. 1 Виконання, оформлення та захист випускних робіт бакалавра та атестаційних робіт магістра / С. О. Субботін, С. К. Корнієнко, А. О. Олійник, С.М. Сердюк, В.М. Льовкін та ін. – Запоріжжя : ЗНТУ, 2018. – 133 с.

25. Richards, M. Fundamentals of Software Architecture [Текст] : An Engineering Approach / Mark Richards, Neal Ford. – O'Reilly Media, 2020. – 419 p.

26. Sommerville, I. Software Engineering [Текст] / I. Sommerville. – Addison-Wesley, 2017. – 790 p.

ДОДАТОК А
ІНДИВІДУАЛЬНІ ЗАВДАННЯ

1. Програмне забезпечення підтримки роботи велопрокату. Менеджери визначають базу велосипедів та велостанцій. Менеджери фіксують видачу, повернення велосипедів на відповідних велостанціях, їхній стан. Менеджер фіксує випадки, коли велосипед стає недоступним (здається в ремонт або вилучається). Звичайні користувачі можуть взяти велосипед на прокат (з наявного переліку велосипедів за кожною велостанцією), переглянути історію використання велосервісу. Розрахунок суми коштів за користування велосипедом виконується автоматизовано під час повернення велосипеду.

2. Програмне забезпечення ведення медичних карток пацієнтів. Пацієнти можуть переглядати тільки власні медичні картки і оформлювати договір з лікарем. Лікар може вносити всю необхідну інформацію до медичних карток власних пацієнтів, формувати рецепти на ліки, вносити результати медичних досліджень (передбачити декілька). Аптекар може отримати доступ до всіх рецептів та продати ліки за відповідним рецептом.

3. Програмне забезпечення підтримки процесу оренди житла. Власники житла можуть розміщувати пропозиції надання житла в оренду, визначаючи період часу, протягом якого пропозиція доступна, знімаючи пропозиції в інший період часу, визначаючи мінімальну та максимальну тривалість оренди. Клієнти можуть виконувати пошук житла за заданими параметрами, бронювати житло, залишати відгук. Рішення про здавання в оренду житла конкретному клієнту приймає власник на основі поданої заявки на бронювання.

4. Програмне забезпечення для організації замовлень в ресторані. Система повинна забезпечувати резервування столів у ресторані клієнтами. Оформлення замовлень виконується офіціантами з зазначенням всієї відповідної інформації. Кухарі отримують інформацію про замовлення, виконання яких очікується. Будь-який користувач може відстежувати стан будь-якого замовлення.

5. Програмне забезпечення підтримки процесу надання освітніх послуг. Для кожної дисципліни викладач визначає набір лекцій, які можуть прочитати студенти в текстовому вигляді. Для кожної лекції визначається період часу, протягом якого вона доступна студентам.

Для кожної дисципліни викладач визначає завершальний тест, який проводиться після лекцій та складається з заданого переліку запитань, для кожного з яких задано одну або більше правильних відповідей. Студенти можуть підписуватися на навчання за дисциплінами, після чого отримують доступ до лекцій та складають тест, що визначає успішність навчання.

6. Програмне забезпечення підтримки процесу експрес-доставки. Користувачам надається можливість оформлення замовлень на доставку посилок, відстеження поточного стану посилки, визначення параметрів доставки. Користувач може переглядати перелік всіх власних посилок (як відправника, як отримувача окремо). Оператори можуть змінювати стан замовлення, вносити дані про сплату за послуги (користувач може відмовитись від отримання посилки, якщо вона зіпсована). Якщо посилка доставлена пізніше критичної дати, визначеної користувачем, то вартість послуги зменшується.

7. Програмне забезпечення планування завдань. Користувач має можливість задавати завдання, визначаючи для кожного з них час початку та завершення, відсоток виконання. Для кожного завдання може встановлюватися пріоритет, група завдань, а також ресурси, необхідні для виконання даного завдання. Користувач має можливість переглядати завершені завдання, незавершені, обмежувати перегляд відсотком виконання, виконувати пошук, переглядати завдання за конкретну дату.

8. Програмне забезпечення управління нотатками. Користувач може формувати нотатки: задавати для них текст, назву, теги, колір. Для кожної нотатки може прив'язуватися дата. Програма повинна надавати календар, за яким можна переглядати нотатки, прив'язані до конкретних дат. Користувач може переглядати тільки власні нотатки. Нотатки можна архівувати та відновлювати з архіву.

9. Програмне забезпечення для розміщення фотографій. Користувачі можуть розміщувати власні фотографії, виконуючи їх опис, відправляти їх в архів, обмежувати доступ для перегляду, переглядати фотографії інших користувачів (шляхом пошуку за фотографіями або за власниками), оцінювати їх, коментувати, переглядати коментарі та оцінки власних фотографій.

10. Програмне забезпечення підтримки спільних поїздок автомобілем. Власники автомобілів визначають маршрут поїздки, дати, кількість наявних місць, вартість тощо. Користувачі можуть бронюва-

ти місця, залишати відгуки про водіїв і переглядати відгуки інших користувачів, передивлятися історію поїздок водіїв. Відповідно водії можуть переглядати історію поїздок користувачів, що намагаються забронювати місце.

11. Програмне забезпечення замовлення побутових послуг. Користувачі можуть визначати власні побутові послуги, період їх доступності, вартість, географічну зону їх доступності, можливість одночасного її виконання в різних місцях. Замовники послуг можуть виконувати пошук, задаючи власне розташування, а також використовувати додаткові фільтри результатів, оформлювати замовлення послуг та змінювати замовлення.

12. Програмне забезпечення планування та обчислення бюджету поїздки. Користувач може планувати бюджет поїздки, визначаючи витрати на проживання, проїзд, їжу, музеї, інші розваги тощо. У процесі поїздки всі сплачені суми мають фіксуватися. Програма має надавати можливість переглядати відхилення між запланованим бюджетом та фактичними результатами. Для групової поїздки суми можуть вказуватися для кожного учасника або на всіх загалом за деякими статтями витрат (у такому випадку загальна сума розбивається порівну на всіх учасників). Користувач може мати доступ тільки до власного рахунка.

13. Програмне забезпечення обміну речами. Користувачі можуть визначати власні пропозиції наявних речей і для кожної речі визначати речі, на які вони згідні поміняти її. Якщо на річ, яку виставляє користувач є пропозиція обміну, то в процесі її оформлення має визначатися перелік таких пропозицій обміну, з яких можна обрати одну для реалізації.

14. Програмне забезпечення підтримки аукціону товарів. Продавці виставляють на аукціон товари, визначають граничну вартість та термін проведення аукціону. Учасники аукціонів можуть визначати власну суму коштів, які вони готові сплатити за даний товар або викупати товари за граничну ціну. Якщо встановлена ціна менше граничної, то користувач може змінити її на наступних етапах. Перемагає в такому випадку заявка, в якій запропоновано найбільшу суму.

15. Програмне забезпечення продажу квитків на автобус. Для кожного автобуса визначається його маршрут слідування, вартість проїзду, доступні місця тощо. Користувач задає свій маршрут для виконання пошуку або визначає автобус за номером, після чого отримує

дані про наявні автобуси та вільні місця, з яких може обрати необхідні для бронювання. Користувач може переглядати всі виконані ним бронювання, а також скасовувати їх та вносити зміни.

16. Програмне забезпечення оренди автомобілів. Адміністратори визначають перелік наявних автомобілів, їх вартість тощо, а менеджери визначають їх поточне розташування. Орендар може виконувати пошук за необхідними параметрами (місто відправлення, дати, вартість, клас або марка автомобіля тощо), брати авто в оренду. Вартість послуги розраховується автоматизовано перед початком оренди та перераховується після повернення автомобіля (фіксується менеджерами).

17. Програмне забезпечення міської доставки їжі з закладів громадського харчування. Користувачам надається перелік доступних закладів та страви, продукти (разом з описом, назвою, зображенням, вартістю за одиницю), які в них можна замовити. Інформація про страви, продукти вводиться менеджерами відповідного закладу. Менеджер, кур'єр може вносити дані про роботу над замовленням (стан). Користувач може через програму підтверджувати отримання замовлення, а також створювати та модифікувати всі власні замовлення (до зміни стану на прийнятий).

18. Програмне забезпечення рецензування телесеріалів. Програма містить дані про кожен телесеріал (назва, опис, зображення, набір жанрів, режисер, перелік акторів, вартість виробництва тощо), його сезони, а також кожен епізод. Телесеріали можна переглядати за рейтингом, жанрами, режисерами. Користувачі можуть залишати рецензії (текстові) та виставляти оцінки. Інші користувачі можуть оцінювати ці рецензії та залишати відгук на них. В результаті також формується перелік рецензентів за їх оцінками.

19. Програмне забезпечення продажу та придбання нерухомості. Всю нерухомість поділено на приватні будинки, окремі квартири та квартири в новобудовах. Приватні будинки та окремі квартири можуть виставлятися на продаж звичайними користувачами. Квартири в новобудовах можуть виставлятися тільки спеціальними менеджерами. Користувачі можуть подавати заявки на перегляд житла, передивлятися стан заявок, а також після перегляду виставляти оцінку. Господарі можуть передивлятися заявки на всі їх пропозиції або на обрані, підтверджувати перегляд або скасовувати, знімати пропозиції (кількість

кімнат, площа, рік будівництва, зображення, планування тощо), змінювати.

20. Програмне забезпечення відслідковування помилок. Тестувальники (і розробники) можуть вносити дані про виявлені помилки (номер, опис, пріоритет тощо). Розробники і менеджери можуть змінювати стан помилок та вносити інформацію про роботу над помилкою. Менеджери можуть призначати відповідальних за усунення помилок. Різні розробники можуть вносити коментарі щодо кожної помилки (обговорення). Можна відслідковувати стан кожної помилки (нова, призначена, розв'язана, закрита), виконуючи пошук за різними параметрами, переглядати перелік помилок за обраним станом, переглядати статистику роботи над усуненням помилок, статистику роботи окремих розробників, тестувальників.

21. Програмне забезпечення організації екскурсій. Автори екскурсій можуть визначати екскурсії (з описом, маршрутом), доступні дати для кожного варіанту, вартість (різну для різних параметрів та різних дат), кількість відвідувачів. Клієнти можуть обирати екскурсії (пошук може виконуватися за місцем, датою або проміжком дат, вартістю), бронювати місця, ставити запитання, отримувати на них відповіді, а також голосувати за відкриття нових екскурсій, пропонуючи їх.

22. Програмне забезпечення для збору коштів на реалізацію власних проєктів. Автори можуть визначати власні проєкти, кошти, необхідні на їх фінансування, період збору коштів, протягом якого користувачі можуть перераховувати кошти на фінансування. Якщо на момент закінчення періоду збору коштів необхідна сума не зібрана, то кошти повертаються користувачам. Якщо необхідна сума зібрана раніше, то автор може закінчити збір коштів. Після виконання проєкту кожен користувач має підтвердити його результати або спростувати (в такому випадку автор отримує штраф).

23. Програмне забезпечення проведення іспитів на отримання водійського посвідчення. Адміністратор може визначати перелік тестових запитань (з зображеннями в тому числі), відповідей та правильні відповіді на кожне запитання. Звичайні користувачі можуть здавати іспит з обмеженнями в часі. Набір запитань для кожного тесту формується випадковим чином. У результаті іспиту користувач має отримати повідомлення про те, чи успішно він склав іспит, та свій загальний результат. Для кожного користувача мають зберігатися результати його іспитів. Менеджери можуть виконувати пошук користувачів та

переглядати їхні результати. У випадку, якщо кількість невдалих спроб перевищила встановлений ліміт, доступ до тестів має бути для нього заблокований на деякий час.

24. Програмне забезпечення керування успішністю академічної групи. Програма повинна надавати можливість відстежувати відвідування занять студентами, їх поточні результати та результати складання сесії. Кожен студент може переглядати тільки власні результати. Викладачі можуть виставляти оцінки загалом за курсом, поточні оцінки (кількість таких оцінок за курс визначається викладачем перед початком курсу) та вносити дані про відвідуваність тільки за власними предметами, але можуть переглядати дані про всіх студентів, а також отримувати статистичні дані про кількість боржників, успішність складання сесії тощо. Менеджери можуть вносити дані про дисципліни, які викладаються протягом семестру, та пов'язувати з ними відповідальних викладачів.

25. Програмне забезпечення для підтримки роботи кав'ярні. Програма дозволяє відслідковувати замовлення бару (формувати, переглядати, виконувати). Клієнти можуть оформлювати замовлення за столиком і за баром. Офіціанти можуть вносити інформацію про стан виконання замовлення з керуванням ними за столиками. Клієнтам надається розрахунок. Адміністратори можуть переглядати час і стан поточного обслуговування клієнтів.