МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра інженерії програмного забезпечення

КУРСОВИЙ ПРОЕКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

з дисципліни: «Моделювання та аналіз програмного забезпечення»

на тему:

«LMS система для закладів загальної середньої освіти»

студента IV курсу групи ІПЗ-19-3
спеціальності 121 «Інженерія програмного забезпечення»
Оверчук Валентини Сергіївни
(прізвище, ім'я та по-батькові)

Керівник С.М. Кравченко
Дата захисту: " ___ " _____ 2023 р.
Національна шкала _____
Кількість балів: _____
Оцінка: ECTS _____

Члени комісії _____ І.І. Сугоняк
(підпис) (прізвище та ініціали)
_____ О.В. Власенко
(підпис) (прізвище та ініціали)
_____ С.М. Кравченко
(підпис) (прізвище та ініціали)

Житомир – 2023

ЗМІСТ

ВСТУП

Розробка програмного забезпечення пов’язана з проблемами якості, вартості та надійності. Деякі програми містять мільйони рядків вихідного коду, які, як очікується, повинні правильно виконуватися в умовах, що змінюються. Одним з найскладніших процесів при розробці ПЗ є пошук і виправлення помилок, що впливають на надійність та якість програмного продукту. Навіть якщо архітектура продукту в цілому вибрана правильно, більш локальні помилки наявні завжди: друкарські помилки, невизначена поведінка, завжди помилкові або істинні умови.

Для створення успішного продукту необхідно мати перш за все чітко описаний об’єкт проектування, побудовані повні, несуперечливі функціональні та інформаційні моделі інформаційних систем.

Модель - абстракція фізичної системи, що розглядається з певної точки зору і представлена на деякій мові, або в графічній формі

Побудова моделі - завдання, яке потребує аналізу і синтезу вихідних даних, теорій, гіпотез та знань фахівців.

Метою даної курсової роботи є дослідження особливостей моделювання та аналізу програмних комплексів, які будуть використанні при розробці програмного продукту «LMS система для закладів загальної середньої освіти».

Завданням на курсову роботу є :

- Аналіз та опис вимог - збір даних про те, що повинна робити система
- Аналіз теоретичних засад моделювання програмного забезпечення
- Фізичне моделювання програмних комплексів
- Методи моделювання функцій та поведінки системи
- Проектування об’єктної структури системи
- Кодогенерація із моделей.

Предметом дослідження даного курсової роботи є можливість застосування CASE – засобів проектування програмного забезпечення.

Об’єктом дослідження в даній курсовій роботі є методи та засоби проектування програмного забезпечення та уніфікації процесу проектування.

1 АНАЛІЗ ВИМОГ КОРИСТУВАЧА ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ

1.1 Технічне завдання на розробку системи

1.Назва системи:

Повне найменування системи: LMS система для закладів загальної середньої освіти.

Коротке найменування системи: Веб-додаток, Веб-сайт, сайт, Програмний комплекс, Система.

2.Призначення системи:

Програмний комплекс призначений для спрощення навчального процесу та взаємодії вчителя та учня. Система передбачає також зручне додавання уроків, можливість проходження навчального матеріалу, перегляд результатів.

3.Вимоги до системи:

Програмний продукт повинен мати наступні функціональні характеристики:

- Доступ до системи за допомогою web-інтерфейсів;
- розгалуження системи на частину для учня (перегляд і проходження навчального матеріалу, перегляд результатів) та частину для вчителя (створення навчального матеріалу, завантаження результатів, перегляд статистики) для покращення швидкості роботи;
- зрозумілий та інтуїтивний дизайн;
- можливість одночасної роботи із системою 100 і більше користувачам;
- наявність 3 ролей користувачів (учень, вчитель і адміністратор).

4.Вимоги до програмного забезпечення:

Програмний комплекс повинен бути побудований на клієнт-серверній архітектурі з використанням веб-технологій, що не вимагають додаткового ліцензування. Серверна частина програмного комплексу має бути реалізована з використанням Node.JS та системи управління базами даних MongoDB. При розробці передбачається використання фреймворку

React JS. Важливою концепцією React є компоненти. Дана абстракція дозволяє збирати великі програми з маленьких «шматочків». Вони являють собою придатні до повторного використання об'єкти. При проектуванні програмного комплексу необхідно застосувати архітектуру Model-View-Controller для розмежування алгоритмічної частини від інтерфейсу та забезпечення спільного використання компонентів.

Клієнтська частина повинна бути побудована на стандартних веб-технологіях, що не вимагають встановлення додаткових компонентів на комп'ютер (не рекомендується використовувати Flash або Java-аплети). Сторінки інтерфейсу генеруються сервером за стандартом HTML5, оформлення здійснюється за допомогою таблиць стилів CSS. Інтерфейс повинен коректно працювати у сучасних веб-браузерах Mozilla FireFox та Google Chrome. Також, є можливим використання готових бібліотек для пришвидшення розробки застосунку та покращення його швидкодії.

5.Вимоги до інтерфейсу:

Взаємодія користувачів із програмним комплексом повинна відбуватися за допомогою веб-браузера. Інтерфейс програмного комплексу повинен бути зручним у використанні та інтуїтивно зрозумілим. Елементи керування та компоненти інтерактивної взаємодії повинні бути виконані у зручній для користувача формі. Введення та виведення даних в систему, введення команд та результат їх виконання повинні виконуватися в інтерактивному режимі.

На інтерфейси користувача накладаються наступні вимоги:

- мінімальний час відгуку системи;
- повідомлення про критичні ситуації системи;
- підтвердження виконання критичних дій в системі;
- будь-яка дію користувача має супроводжуватися результатом;
- відображення процесу завантаження для ресурсомістких завдань.

1.2 Обґрунтування вибору засобів моделювання

UML (англ. Unified Modeling Language) – це мова позначень або побудови діаграм, призначена для визначення, візуалізації і документування моделей зорієнтованих на об’єкти систем програмного забезпечення. UML не є методом розробки, іншими словами, у конструкціях цієї мови не повідомляється про те, що робити першим, а що останнім, і не надається інструкцій щодо побудови вашої системи, але ця мова допомагає вам наочно переглядати компонування системи і полегшує співпрацю з іншими її розробниками.

Для моделювання даної курсової роботи розглянемо нижче наведені засоби UML:

- ArgoUML
- Dia
- StarUML

Зробимо аналіз кожного засобу:

ArgoUML:

– Підтримка UML - UML 1.4

– Генерація вихідних кодів по UML діаграмі - Java, C++, C#, PHP. Можна додати інші мови

– Зворотне проектування вихідних кодів – Java

– Не підтримує бази даних

– Не підтримує багатокористувацький режим роботи

– Є відкритим програмним забезпеченням

Dia:

– Підтримка UML - 2.0

– Генерація вихідних кодів по UML діаграмі - Ada, C, C++, Java, PHP, Python, SQL (за допомогою Dia2Code)

– Зворотне проектування вихідних кодів – Perl, Java, PHP, C++ (за допомогою AutoDia)

– Підтримує роботу з базами даних

– Є відкритим програмним забезпеченням

StarUML:

– Підтримка UML - 2.0

– Перевірка правильності UML діаграм

– Генерація вихідних кодів по UML діаграмі – Ada, Java, C #, C++, на офіційному сайті є доповнення для інших мов

– Зворотне проектування вихідних кодів – наявне

– Підтримка патернів

– Експорт документації в формати: DOC, PPT, TXT, XLS та інші

– Не підтримує роботу з базами даних

– Зручний графічний редактор

– Є відкритим програмним забезпеченням

Після проведення порівнянь засобів UML було обрано «StarUML», завдяки тому, що він підтримує генерацію коду на різні мови програмування, є зручним у використані, а також має безкоштовну версію.

1.3 Аналіз вимог до програмного продукту

Розглянемо вимоги, яким повинен задовільняти програмний продукт, щоб мати можливість виконувати всі покладені на нього фунцій, тим самим задовольнивши всі зазначені стандарти, специфікації та інші формальні документи. Приведені можливості системи, обмеження та показники якості.

Бізнес-вимоги:

1. Основні цілі: проект створюється з метою спрощення навчального процесу та взаємодії вчителя та учня.
2. Представлення проекту: проект буде реалізовано у вигляді сайту, що містить структуровану інформацію по даній тематичній області.

Вимоги зовнішних користувачів (Учень)

1. Можливість переглядати власний профіль, редагувати певні поля

2. Можливість сортування, фільтрації та пошуку матеріалів
3. Можливість проходження навчальних матеріалів
4. Можливість перегляду активних та пройдених матеріалів
5. Можливість перегляду власних досягнень
6. Можливість редагування особистого кабінету

Вимоги зовнішних користувачів (Вчитель)

1. Можливість редагування власного профілю
2. Можливість перегляду профілю учня
3. Можливість створення, редагування та видалення навчальних матеріалів
4. Можливість перегляду результатів проходження матеріалу, як окремого учня, так і цілого класу.
5. Завантаження результатів
6. Можливіст перегляду статистики

Вимоги внутрішних користувачів (Адміністратор):

1. Можливість зміни будь-якої інформації у БД
2. Можливість архівування БД чи окремої таблиці
3. Можливість реєструвати користувачів
4. Можливість надання прав Учня або Вчителя
5. Можливість створення класів
6. Можливість додавання учнів до класів

Характеристика об`єкту комп`ютеризації:

Користувач додатку має достатньо можливостей та зручно розроблений інтерфейс для взаємодії з учнями чи вчителями та адміністрацією.

Функціональні вимоги:

1. Авторизація в системі: В системі повинна бути представленна можливість реєстрації користувача та присвоєння йому відповідної ролі (учень, вчитель, адміністратор).

2. Можливість збереження інформації: Система повина зберігати інформацію і надавати можливість керувати нею.

Нефункціональні вимоги:

Сприйняття

- Час, потрібний для навчання інструментами роботи з інформаційною системою для звичайних користувачів – 2-3 години, а для досвідченних – 1 година
- Час відповіді системи для звичайних запитів не повинення перевищувати 5 секунд, а для більш складних запитів – 10 сек.
- Інтерфейс представлення ІС повинен бути інтуітивно зручним для користувача та не вимагати від нього додаткової підготовки.

2. Надійність
   - Доступність – час, потрібний для обслуговування системи не повинен перевижувати 10% від загального часу роботи.
   - Середній час безперервної роботи – 20 робочіх днів.
   - Максимальна норма помилок та дефектів в роботі системи – 1 помилка на 1000 запитів користувача.
3. Продуктивність
   - Система повинна підтримувати мінімум 100 одночасно працюючих користувачів, пов’язаних з спільною базою данних.
4. Можливість експлуатації
   - Масштабування – система повинна мати можливість збульшувати потужності (продуктивність), зі збільшенням користувачів таким чином, щоб це аж ні як негативно не вдобразилося на її роботі..

Системні вимоги:

1. **Вимоги до середовища виконання:**

   Система повинна задовольняти зазначеним вимогам на комп’ютері в наступній мінімальній комплектації:

   - підтримка браузера *Google Chrome* версії 45+ або

- підтримка браузера *Mozilla Firefox* версії 38+ або
- підтримка браузера *Microsoft Edge* версії 12+ або
- підтримка браузера *Opera* версії *30*+ або
- підтримка браузера *Safari* версії *9*+

2. **Вимоги до СУБД та доступу до даних:**

- У ядрі системи повинна бути представлена документо-орієнтована система управління базами даних: MongoDB

Аналіз вимог користувачів дав підстави для визначення варіантів використання LMS системи для закладів загальної середньої освіти. На рисунку 1.1 зображено побудовану діаграму варіантів використання.

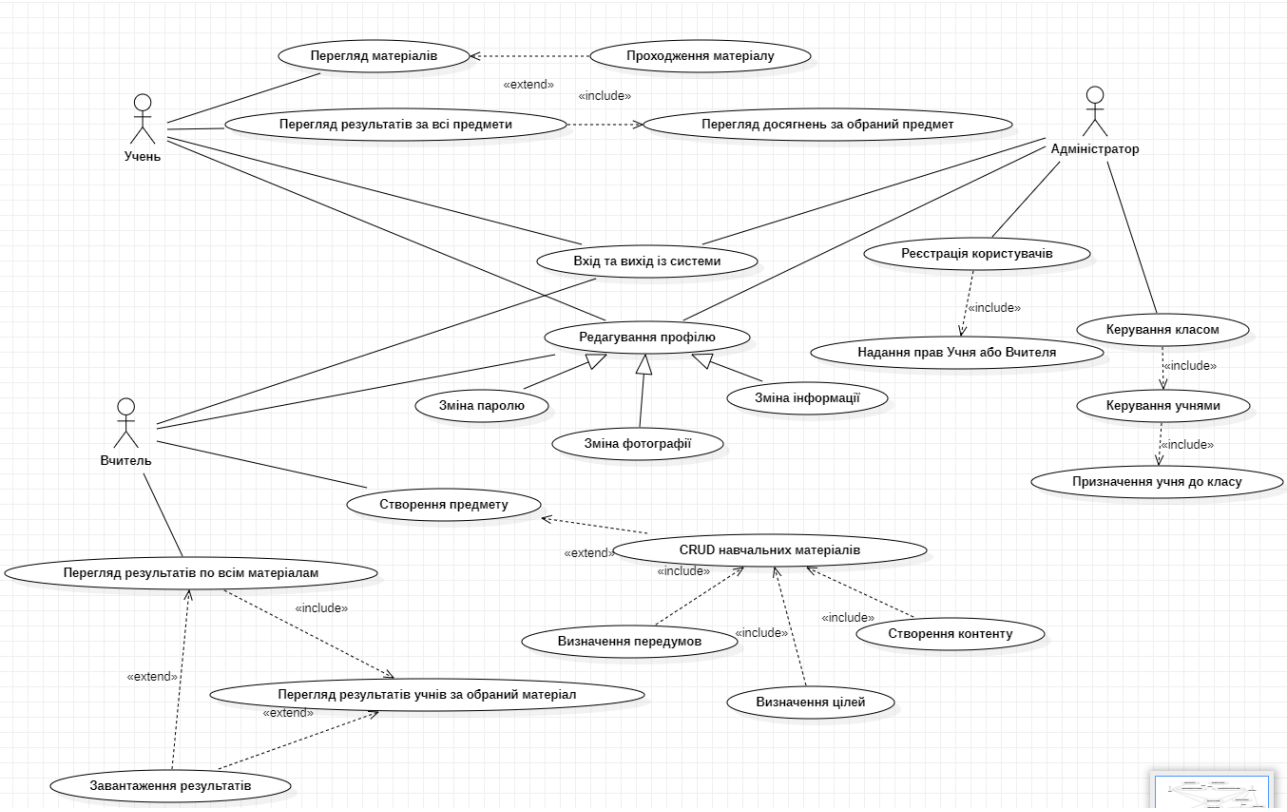

Рисунок 1.1 – Діаграма варіантів використання «LMS система для закладів загальної середньої освіти»

2 РОЗРОБКА МОДЕЛІ ПРОГРАМНОГО КОМПЛЕКСУ НА ЛОГІЧНОМУ РІВНІ

2.1 Алгоритм роботи та стани програмної системи

Під час планування алгоритму роботи та станів програмного додатку спочатку необхідно побудувати діаграму активності(activity diagram).

Діаграма діяльності (Activity diagram) – UML-діаграма, яка дозволяє моделювати послідовності бізнес-процесів або дій, реалізованих методами класів. Зазначені послідовності можуть являти собою альтернативні галузі процесу обробки даних або галузям, які можуть виконуватися паралельно. Діаграми діяльності є аналогом блок-схеми будь-якого алгоритму.

Було створено 3 діаграми активності для можливих сценаріїв в роботі системи.

На рисунку 2.1 зображено діаграму активності та відповідності стані дій для перевірки прав Вчителя при воду і систему.

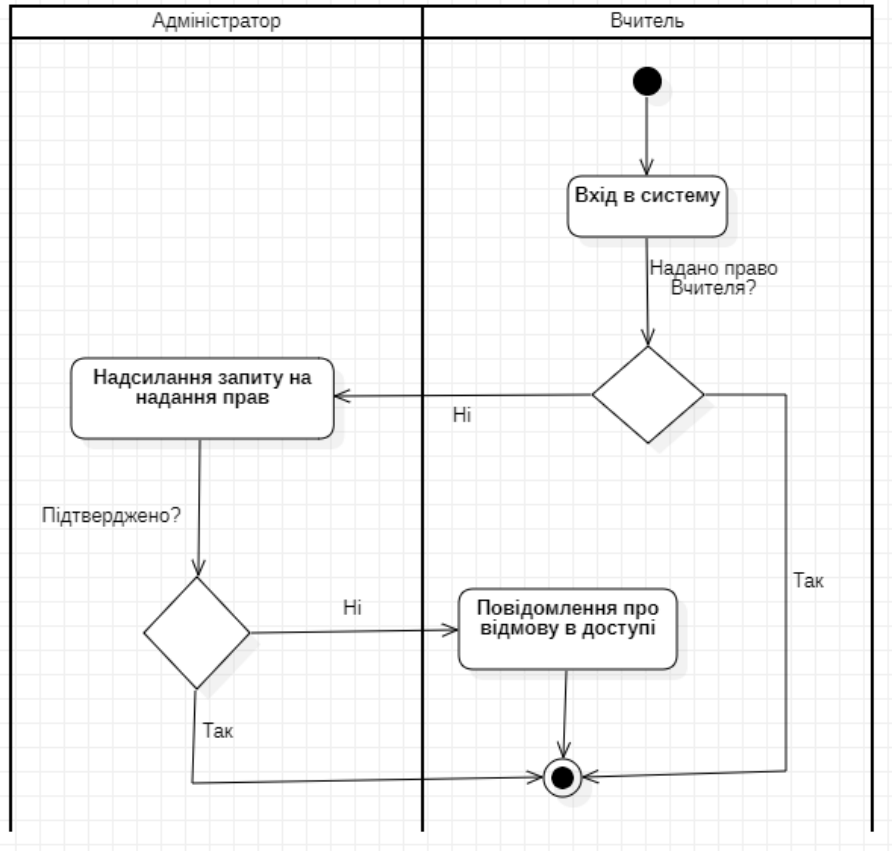

Рисунок 2.1 – Діаграма дій перевірки прав Вчителя

Опис діаграми зображено на рисунку 2.1. На цій діаграмі розміщені дві доріжки, які відповідають ролям:

Вчитель – здійснює вхід у систему. Адмінітратор – обробляє, надає право Вчителя при надсилання запиту до нього.

- Початковий стан - чорний кружок
- Кінцевий стан - чорний кружок в колі
- Рішення - ромби, а саме:
  - Чи надано користувачу право Вчитель?
  - Чи Адміністратор підтверджує запит про надання права Вчителя?
- Округлені прямокутники позначають дії. На діаграмі наявні наступні
  - Вхід у систему
  - Надсилання запиту на надання права
  - Повідомлення про відмову в доступі

На рис. 2.2 наведено діаграму активності та відповідності стані дій для процесу створення матеріалу вчителем та проодження його учнем.

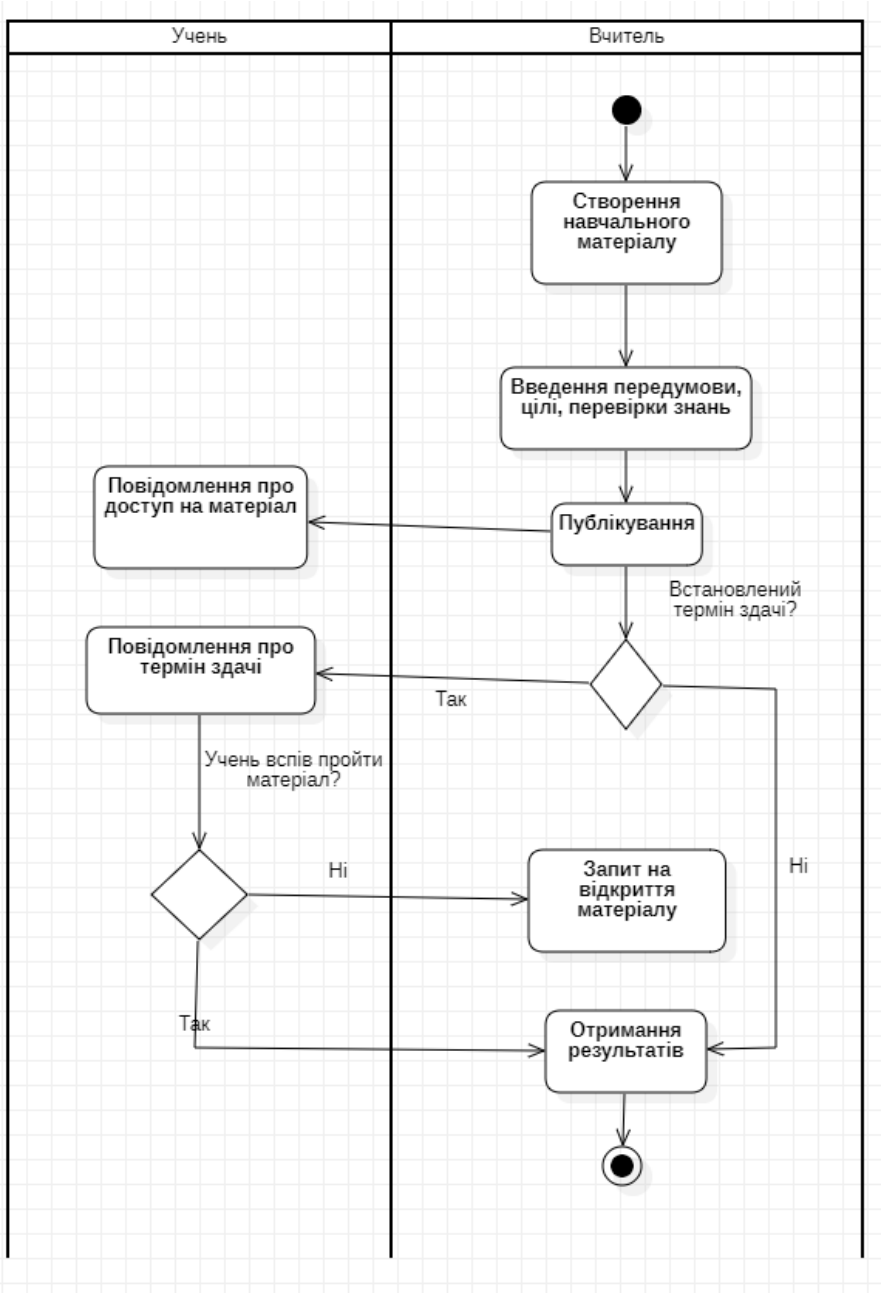

Рисунок 2.2 – Діаграма дій створення та проходження матеріалу

Опис діаграми створення та проходження матеріалу (Рис. 2.2)

- Початковий стан - чорний кружок
- Кінцевий стан - чорний кружок в колі
- Рішення - ромби, а саме:
  - Чи встановлений термін для проходження матеріалу?
  - Чи було пройдено матеріал вчасно?
- Округлені прямокутники позначають дії. На діаграмі наявні наступні
  - Створення навчального матеріалу

- Введення передумови, цілі, перевірки знань
- Публікування матеріалу
- Повідомлення про доступ на матеріал
- Повідомлення про термін здачі
- Запит на відкриття матеріалу
- Отримання результатів

На рис. 2.3 наведено діаграму активності та відповідності стані дій для взаємодії учня та вчителя.

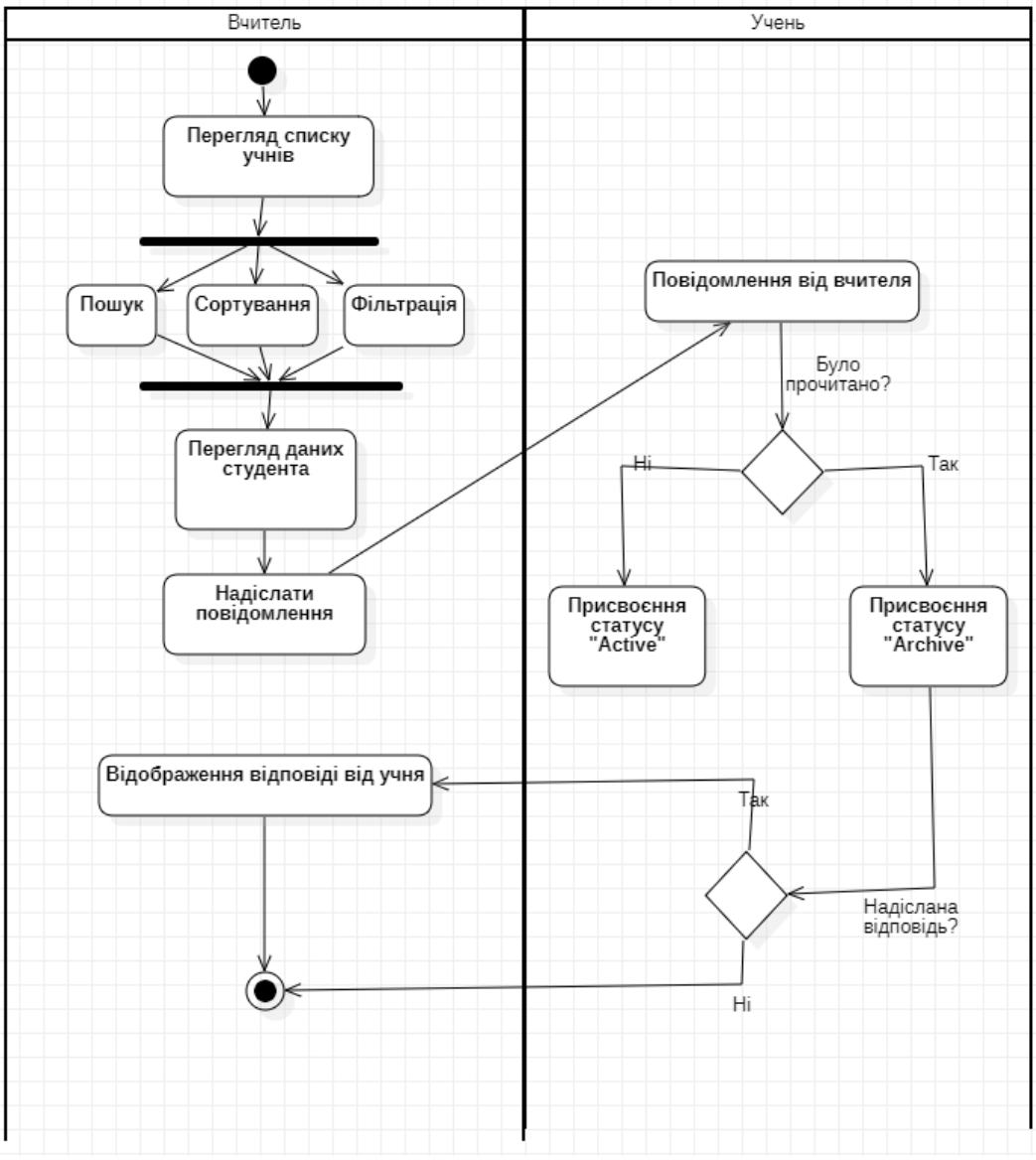

Рисунок 2.3 – Діаграма дій взаємодії учня та вчителя

Останньою діаграмою є діаграма діяльності взаємодії учня та вчителя. Тут також розміщені дві доріжки, як і в попередній діаграмі.

– На діаграмі наявні наступні дії:

- Перегляд списку учнів
- Пошук, сортування, фільтрація учнів
- Перегляд даних студента
- Надсилання повідомлення
- Присвоєння статусу повідомлення: “Active” або “Archive”

2.2 Об’єктно-орієнтована модель системи

Діаграма класів (Class diagram) - основна діаграма для створення коду додатка. За допомогою діаграми класів створюється внутрішня структура системи, описується спадкування й взаємне положення класів друг щодо друга. Тут описується логічне представлення системи. Саме логічне, тому що класи - це лише заготовки, на основі яких потім будуть визначені фізичні об'єкти. Клас в мові UML служить для позначення множини об’єктів, які мають однакову структуру, поведінку і відносини з об’єктами інших класів. Графічно клас зображується у вигляді прямокутника, який додатково може бути розділений горизонтальними лініями на розділи або секції. У цих розділах можуть зазначатися ім’я класу, атрибути (змінні) та операції (методи).

На рисунку 2.2 зображено побудовану діаграму класів.

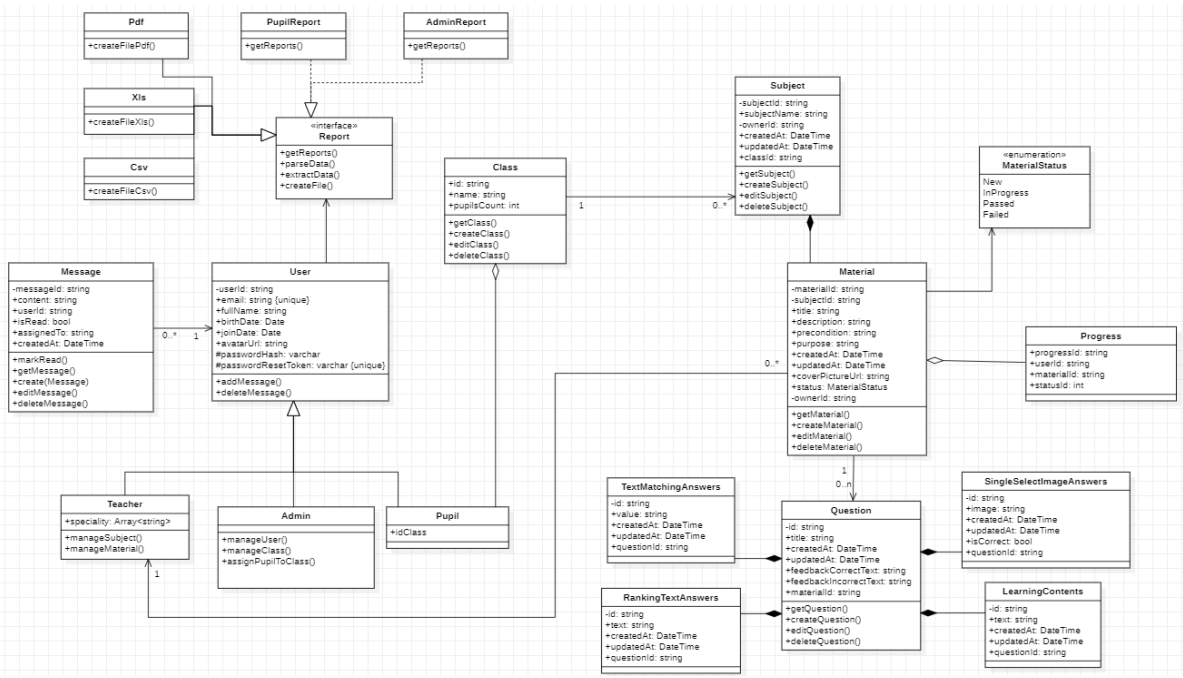

Рисунок 2.2 – Діаграма класів з патернами проектування

Опис класів, а також їх полів та методів (у класах використовується англомовна назва полів та методів з використанням так званого верблюжого регістру - camelCase):

- Клас User є абстрактним класом, він батьківський клас для Teacher, Pupil та Admin. Має такі властивості як: password, username – необхідні для авторизації у систему; персональні дані – firstname, lastname, email, image. Наявні методи для управління коментарями (addMessage, deleteMessage);
- Класи Pupil, Teacher та Admin наслідують клас User та служать для розмежування можливостей облікових записів;
- Клас Message створений для спілкування учня та вчителя. Містить код користувача, вміст коментаря, дату створення та статус коментаря (активний чи заархівований).
- Інтерфейс Report містить методи для роботи з звітами спільні для усіх облікових записів, від нього реалізуються два класи PupilReport та TeacherReport які розрізняють звіти для облікових записів.
- Класи Pdf, Xls, Csv служать для експорту звітів у однойменні формати документів.
- Клас Class, що описує всю необхідну інформацію про клас: код класу, назву та кількість учнів. Також методи для роботи з класом (додавання, редагування, видалення та отримання класу).
- Клас Question містить інформацію про запитання та реалізує методи, що відповідають за отримання запитань та роботу з ними(створення, редагування та видалення запитань).
- Класи TextMatchingAnswers, RankingTextAnswers, SingleSelectImageAnswers та LearningContents відповідають за варіанти запитань для перевірки знань. Зв’язок з класом Question реалізований у вигляді композиції, а отже при видаленні запитання будуть видалені і варіанти.

– Клас Material містить інформацію про навчальний матеріал та реалізує методи, що відповідають за отримання їх та роботу з ними(створення, редагування та видалення матеріалів).

– Клас Progress зберігає прогрес пройдених матеріалів, зв’язок з класом Material реалізований у вигляді агрегації, отже при видаленні матеріалу, прогрес не видалиться.

– Клас Subject створений для опису предмета. Зберігає наступну інформацію: назву премета, код автора, дату створення та останнього редагування. Також створені методи роботи з предметом(отримання, створення, видалення та редагування предмета).

Як було згадано раніше у проектуванні були використані патерни. Рішення багатьох проблем, що часто зустрічаються, вже існують, вони називаються “патернами проектування”. Вони можуть прискорити процес розробки. Патерни проектування створюються для виправлення відомих проблем, тобто їх появі можна запобігти до того, як вони стануть видимими в процесі реалізації.

На відміну від готових функцій чи бібліотек, патерн не можна просто взяти й скопіювати в програму. Патерн являє собою не якийсь конкретний код, а загальний принцип вирішення певної проблеми, який майже завжди треба підлаштовувати для потреб тієї чи іншої програми.

Ви можете цілком успішно працювати, не знаючи жодного патерну. Більше того, ви могли вже не раз реалізувати який-небудь з патернів, навіть не підозрюючи про це. Отже, навіщо ж знати патерни?

- Перевірені рішення;
- Стандартизація коду;
- Загальний словник програмістів.

Патерни відрізняються за рівнем складності, деталізації та охоплення проектованої системи. Розглянемо основні групи патернів:

- Породжуючі патерни піклуються про гнучке створення об’єктів без внесення в програму зайвих залежностей;

- Структурні патерни показують різні способи побудови зв’язків між об’єктами;
- Поведінкові патерни піклуються про ефективну комунікацію між об’єктами.

Ми не будемо розглядати усі існуючі патерни, а зупинимось на тих які були використані:

1. Прототип – це породжуючий патерн проектування, що дає змогу копіювати об’єкти, не вдаючись у подробиці їхньої реалізації. Клас User є прототипом до класів Teacher, Admin та Pupil. Реалізацію патерну представлено на рис.2.3

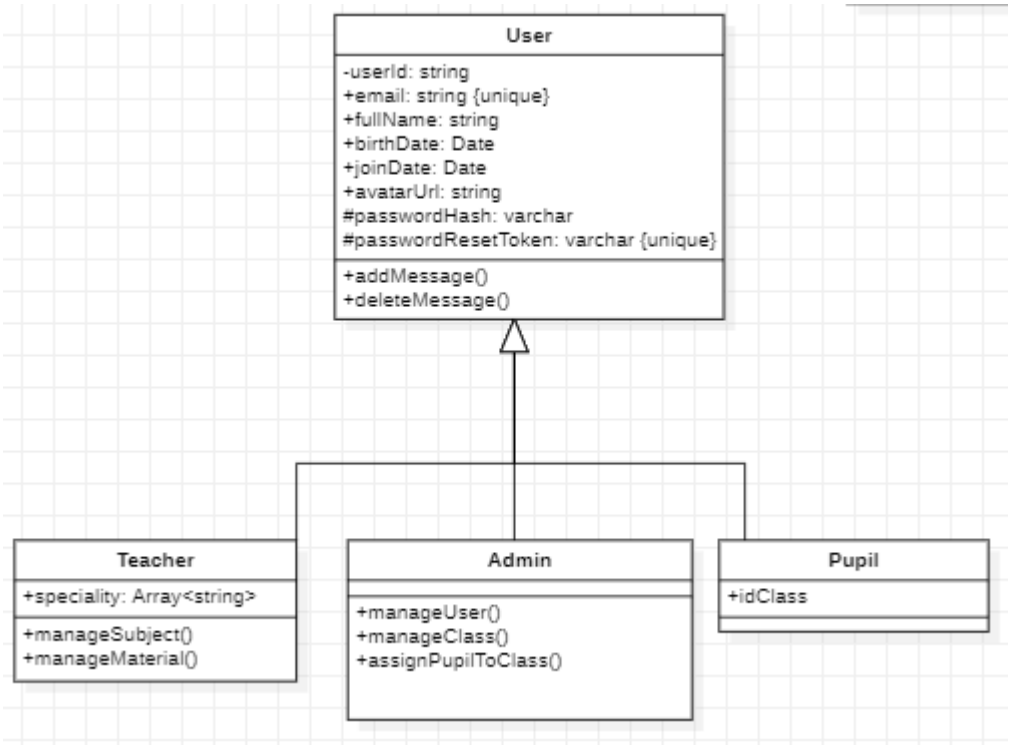

Рисунок 2.3 – Реалізація патерну “ Прототип”

2. Шаблонний метод – це поведінковий патерн проектування, який визначає кістяк алгоритму, перекладаючи відповідальність за деякі його кроки на підкласи. У інтерфейсі було виділено спільні методи для усіх звітів, та унаслідували його для різних типів документів. Реалізацію патерну представлено на рис.2.4

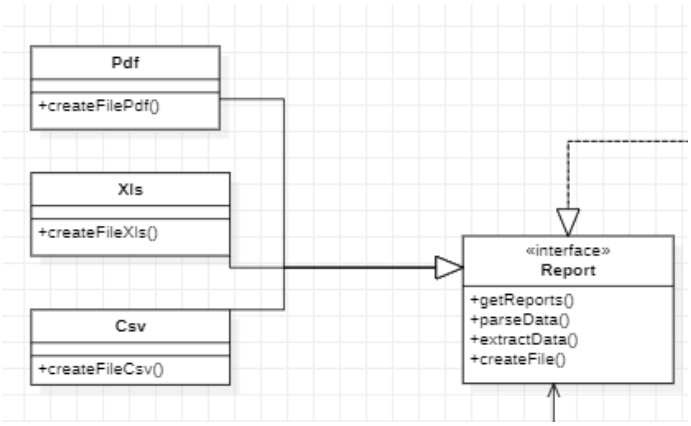

Рисунок 2.4 – Реалізація патерну “ Шаблонний метод”

3.Фабричний метод – це породжуючий патерн проектування, який визначає загальний інтерфейс для створення об’єктів у суперкласі, дозволяючи підкласам змінювати тип створюваних об’єктів. Було використано цей шаблон проектування для генерації звітів, які для учня та вчителя є різними, але успадковуються від одного інтерфейсу. Реалізацію патерну представлено на рис.2.5

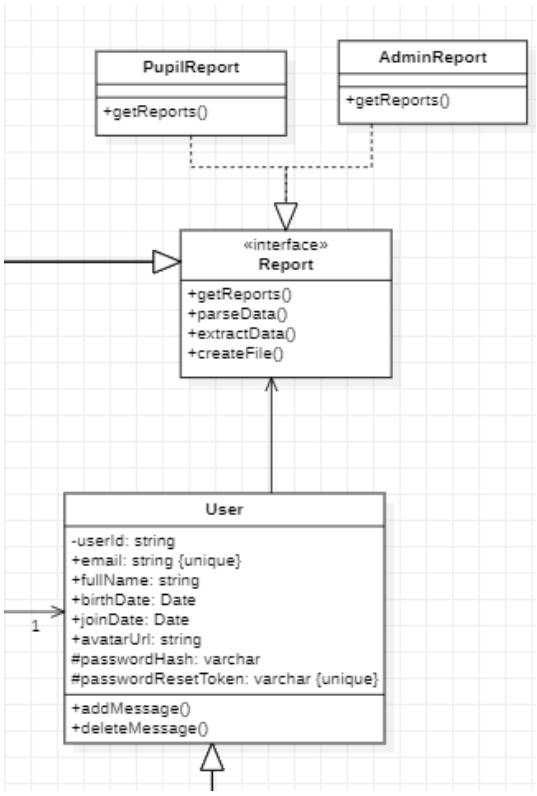

Рисунок 2.4 – Реалізація патерну “ Фабричний метод”

2.3 Взаємодія об’єктів системи

Діаграма послідовності (Sequence diagram) – діаграма, яка відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність відправлених повідомлень. Діаграми послідовностей можна використовувати для уточнення діаграм прецедентів, більш детального опису логіки сценаріїв використання. Це відмінний засіб документування проекту з точки зору сценаріїв використання. Діаграми послідовностей зазвичай містять об'єкти, які взаємодіють у рамках сценарію, повідомлення, якими вони обмінюються, і які повертаються результати, що пов'язані з повідомленнями.

На рис. 2.5 наведено діаграму послідовності для процесу створення та проходження матеріалу.

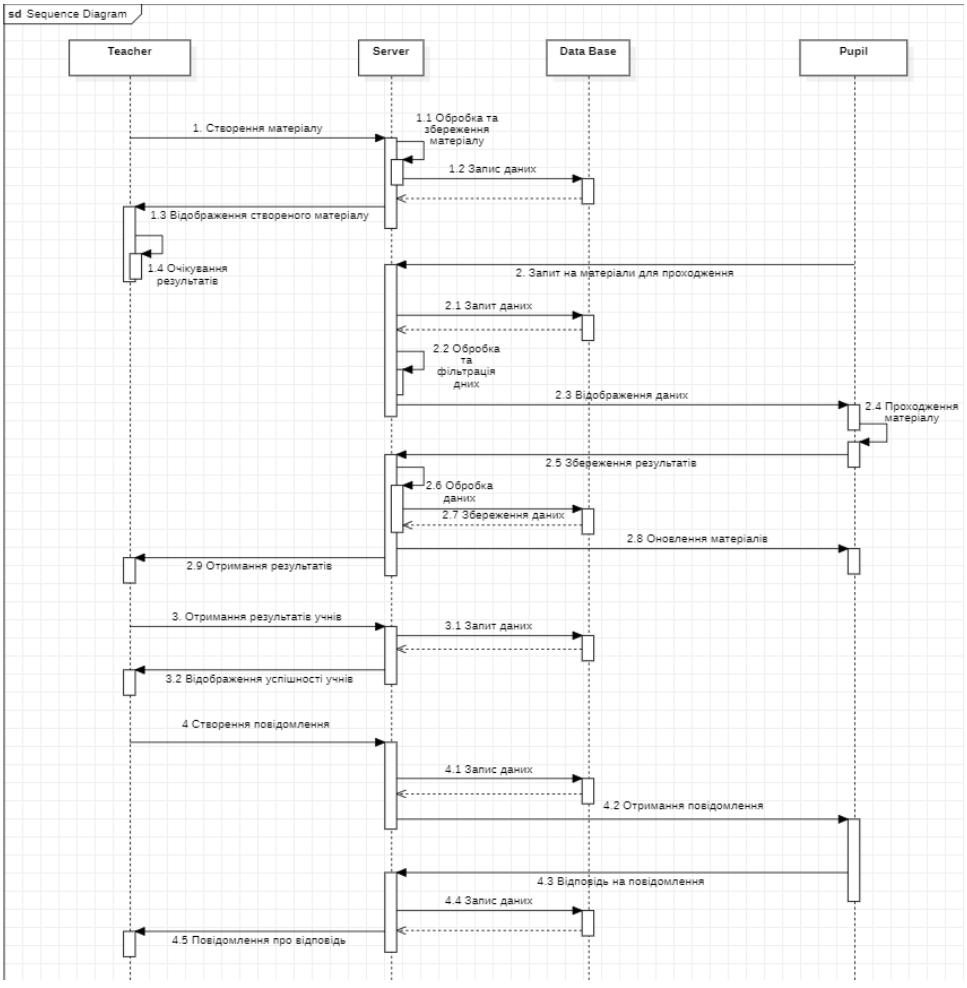

Рисунок 2.5 – Діаграма послідовності процесу створення та проходження матеріалу

3 ФІЗИЧНА МОДЕЛЬ ТА ПРОТОТИП ПРОГРАМНОГО КОМПЛЕКСУ

3.1 Взаємодія компонентів системи

Діаграма компонентів – статична структурна діаграма, що показує розбиття програмної системи на структурні компоненти та зв'язки (залежності) між компонентами. Як фізичні компоненти можуть виступати файли, бібліотеки, модулі, виконувані файли, пакети.

На рисунку 3.1 зображено діаграму компонентів.

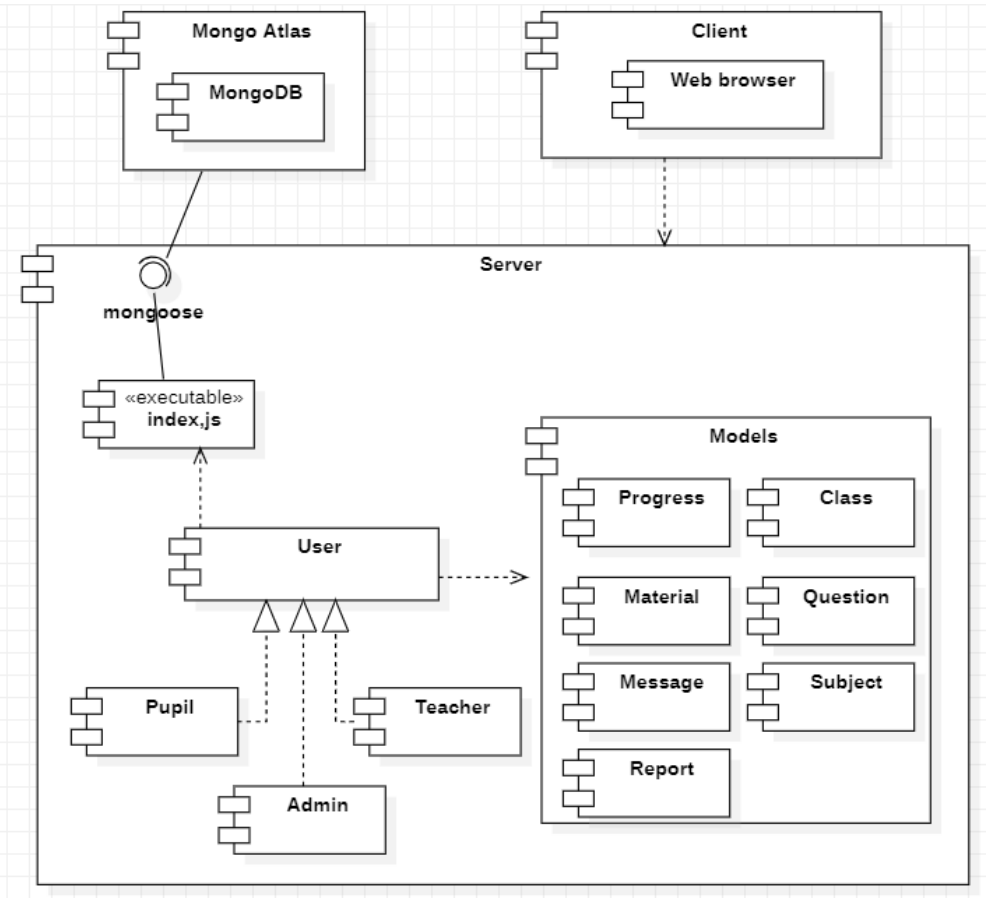

Рисунок 3.1 – Діаграма компонентів

На діаграмі зображено такі елементи:

- Компоненти – фізично існуюча частина системи, яка забезпечує реалізацію класів і відносин, а також функціональної поведінки модельованої програмної системи.
- На цій діаграмі це:
  - Models – компоненти, які є модулями в програмному коді:
  - Залежності

- Відношення залежності
- Mongo Atlas – хмарний сервер бази даних:
  - Базу даних MongoDB;
- Client – пристрій користувача з якого він може користуватись додатком:
  - Веб-браузер.

3.2 Архітектура програмного комплексу та його розгортання

Діаграма розгортання – це діаграма на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонентів. Діаграма розгортання відображає робочі екземпляри компонентів, а діаграма компонентів, натомість, відображає зв'язки між типами компонентів.

Діаграма розгортання в UML моделює фізичне розгортання артефактів на вузлах. Вузли представляються, як прямокутні паралелепіпеди з артефактами, розташованими в них, зображеними у вигляді прямокутників. Вузли можуть мати підвузли, які представляються, як вкладені прямокутні паралелепіпеди. Один вузол діаграми розгортання може концептуально представляти безліч фізичних вузлів, таких, як кластер серверів баз даних.

Діаграма розгортання (Рисунок 3.2) має наступну структуру:

Вузли:

- Client – пристрій користувача
- Web Server – сервер, що обробляє запити користувачів

MongoDb Atlas Server – хмарний сервер бази даних

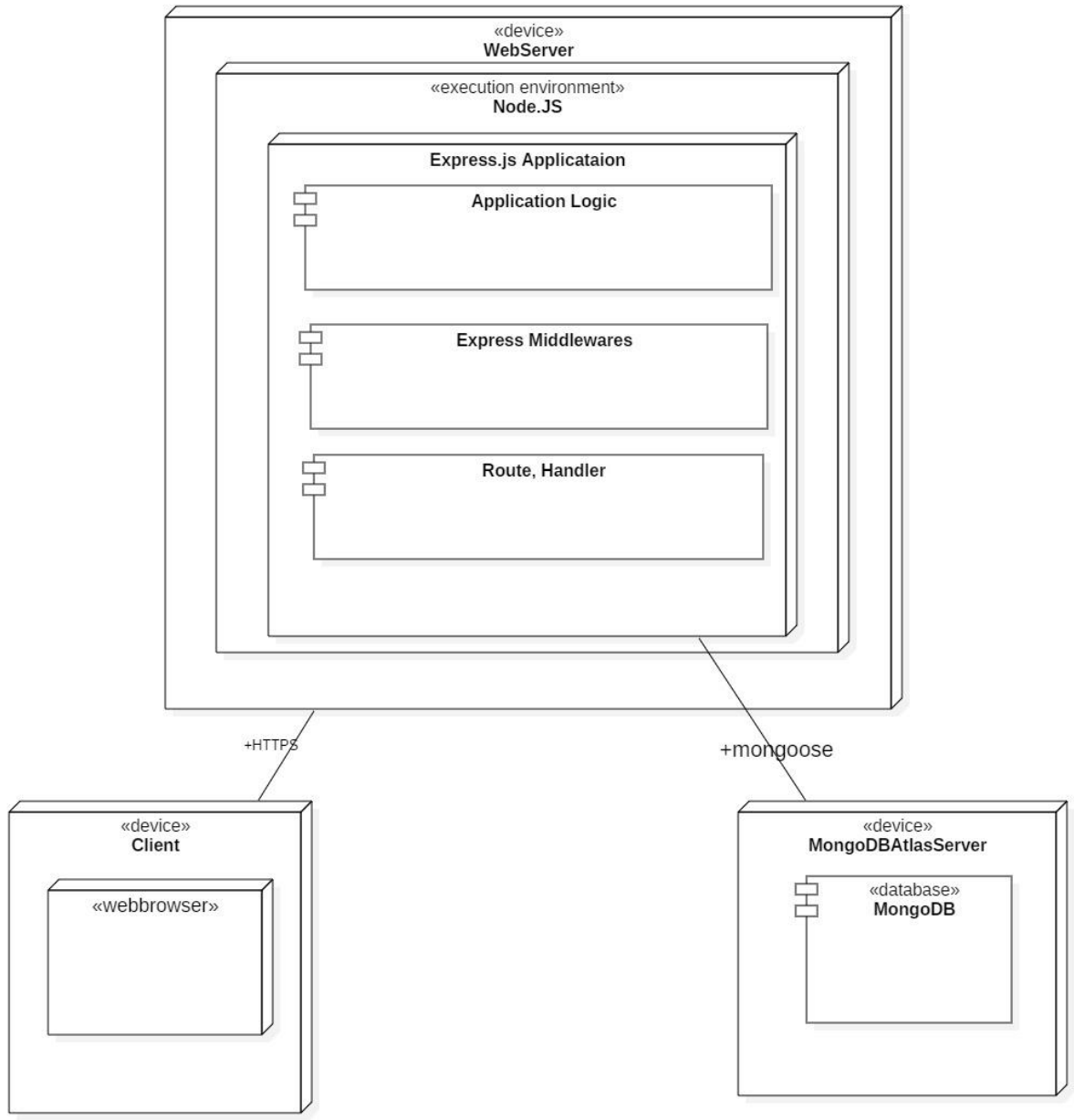

Рисунок 3.2 – Діаграма розгортання

3.3 Генерування програмного коду для прототипу програми

Під час вибору засобів моделювання важливим елементом у редакторі UML-діаграм була наявність кодогенерації.

Для кодогенерації потрібна діаграма класів та розширення для мови програмування. Додаток буде розроблятися на мові програмування JavaScript, але розширення для цієї мови в starUML немає, тому для прикладу була вибрана мова C#.

Для генерації коду C# у StarUML необхідно виконати такий порядок дій:

1. Відкрити файл з створеною раніше діаграмою класів;
2. Перейти у StarUML – Tools – Extension Manager, знайти та встановити розширення C#;
3. Перезавантажити StarUML;
4. Перейти у StarUML – Tools – C# – Generate code та обрати модель, код якої потрібно згенерувати
5. Згенерований код з’явиться у папці яку було обрано.

Згенеровані класи системи зображені на рисунку 3.3, а лістинг коду всіх класів наведено в додатку А.

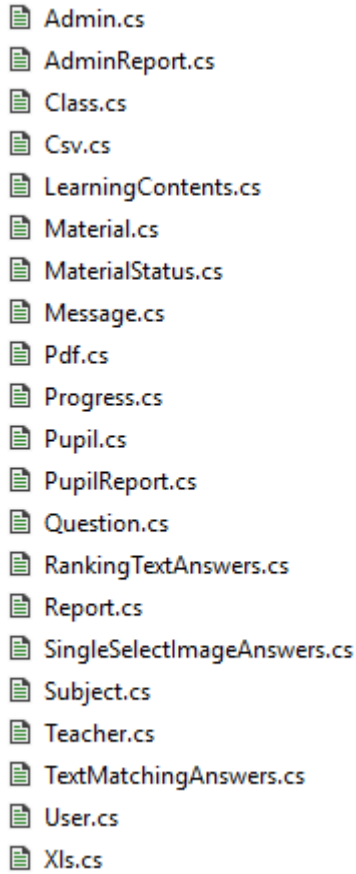

Рисунок 3.3 – Файли класів, які було згенеровано

На основі спроектованої системи були розроблені прототипи кінцевого додатку за допомогою графічного редактора Figma.

На рисунках 3.4–3.5 зображені розроблені прототипи додатку.

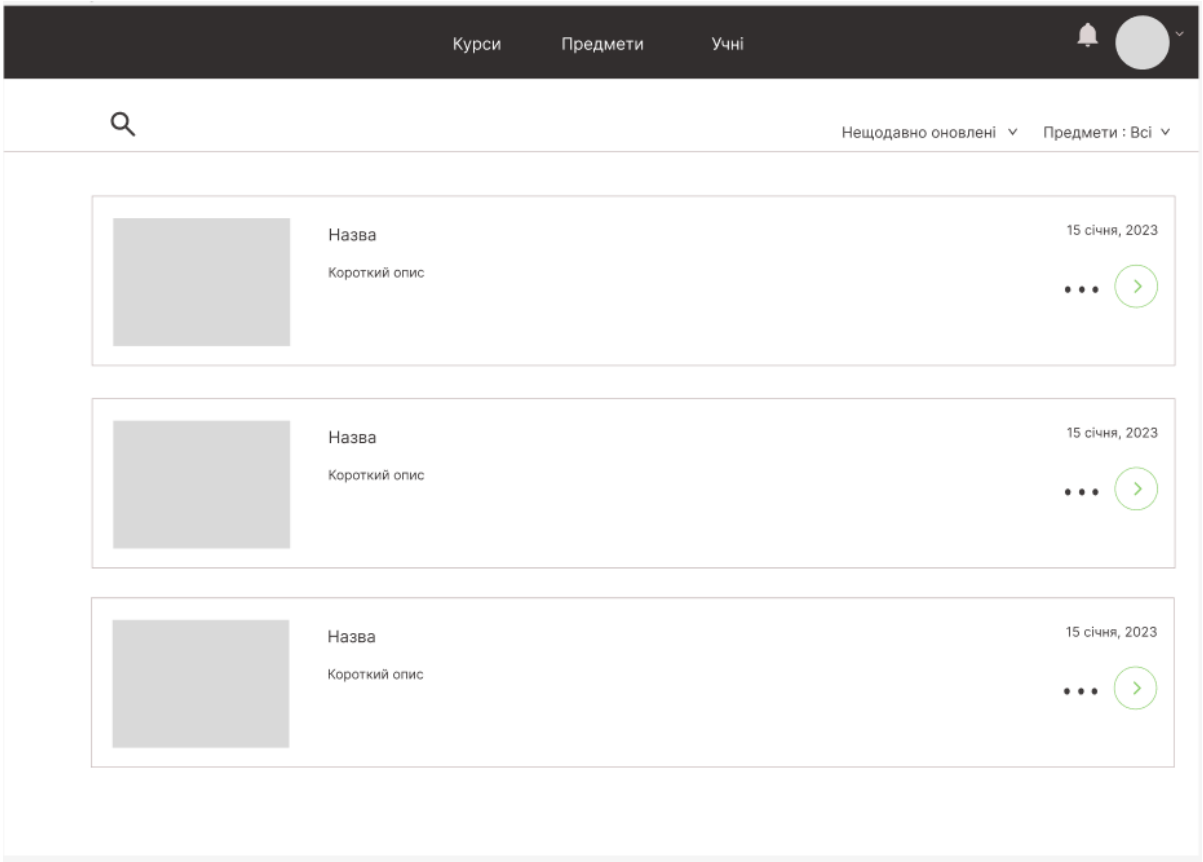

Рисунок 3.4 – Прототип головної сторінки для Вчителя

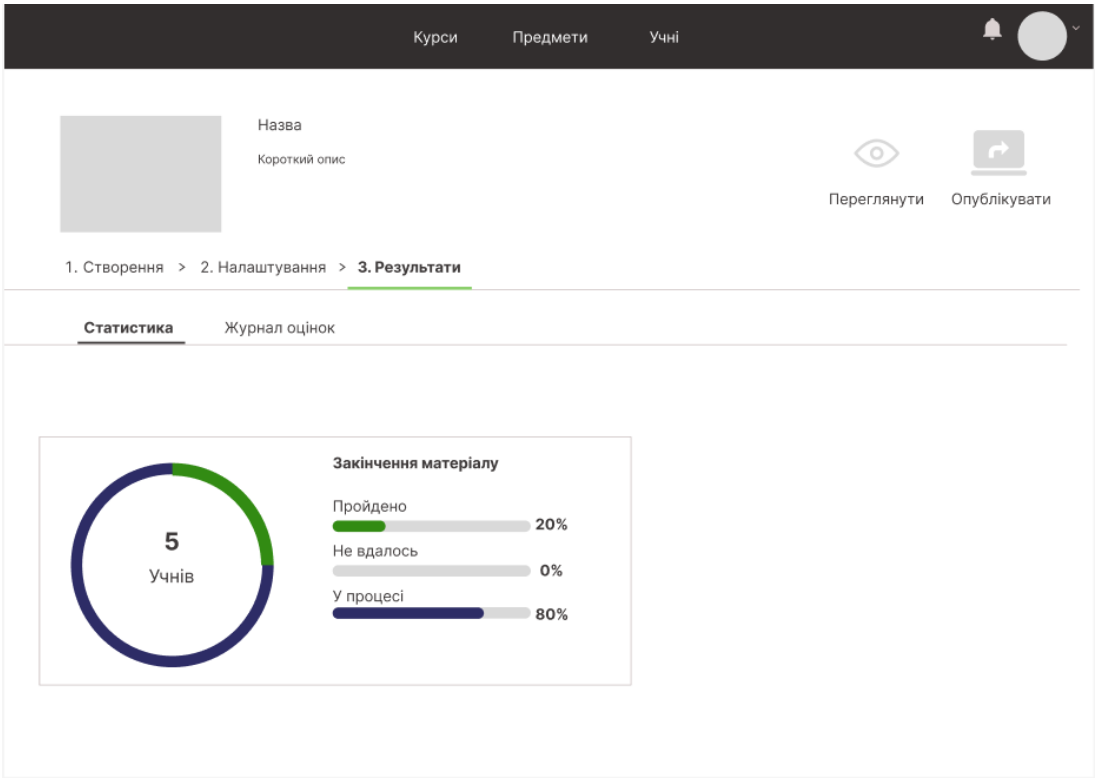

Рисунок 3.5 – Прототип сторінки результатів по матеріалу

ВИСНОВКИ

Під час написання першого розділу було проведено аналіз вимог, щодо кінцевого програмного продукту. Проведено аналіз серед засобів моделювання та обрано найбільш відповідний під необхідні цілі. Також було виконано постановку задачі.

У другому розділі було розроблено алгоритм роботи програми. Були розглянуті та побудовані діаграма діяльності, діаграма класів, діаграма послідовності. Також були обрані патерни проектування для забезпечення кращої розробки додатку.

У третьому розділі було проаналізовано взаємодію між всіма компонентами додатку. Побудовано діаграму компонентів та діаграму розгортання. Також було згенеровано програмний код, та створено прототип веб-додатку.

В результаті виконання курсової роботи було створено модель LMS системи для закладів загальної середньої освіти для подальшої розробки програмної системи. В ході виконання курсової роботи було створено UML-діаграми. Також досліджено особливості моделювання та аналізу програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Графічний редактор StarUML [Електронний ресурс] – Режим доступу до ресурсу: https://staruml.io.
2. Графічний редактор Diagrams.net [Електронний ресурс] – Режим доступу до ресурсу: https://app.diagrams.net/.
3. Графічний редактор Microsoft Visio [Електронний ресурс] – Режим доступу до ресурсу: https://www.microsoft.com/en-us/microsoft-365/visio/flowchart-software.
4. StarUML Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.staruml.io.
5. Патерн Прототип [Електронний ресурс] – Режим доступу до ресурсу: https://refactoring.guru/uk/design-patterns/prototype.
6. Патерн Шаблонний метод [Електронний ресурс] – Режим доступу до ресурсу: https://refactoring.guru/uk/design-patterns/template-method.
7. Патерн Фабричний метод [Електронний ресурс] – Режим доступу до ресурсу: https://refactoring.guru/uk/design-patterns/factory-method.
8. Class Diagrams [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/docs/en/rsm/7.5.0?topic=structure-class-diagrams.
9. Relationships in class diagrams [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/docs/en/rsm/7.5.0?topic=structure-class-diagrams.
10. Sequence Diagram [Електронний ресурс] – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/.
11. Графічний редактор Figma [Електронний ресурс] – Режим доступу до ресурсу: https://www.figma.com/.
12. Class diagram [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Class_diagram.

ДОДАТКИ

Додаток А

Admin.cs

Лістинг програми:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Admin : User {

    public Admin() {
    }

    public void manageUser() {
        // TODO implement here
    }

    public void Operation1() {
        // TODO implement here
    }

    public void Operation2() {
        // TODO implement here
    }

    public void manageClass() {
        // TODO implement here
    }

    public void assignPupilToClass() {
        // TODO implement here
    }

}
```

AdminReport.cs

Лістинг програми:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class AdminReport : Report {

    public AdminReport() {
    }

    public void getReports() {
        // TODO implement here
    }

}
```

Class.cs

Лістинг програми:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Class {

    public Class() {
    }

    public string id;

    public string name;

    public int pupilsCount;

    public void getClass() {
        // TODO implement here
    }

    public void createClass() {
        // TODO implement here
    }

    public void editClass() {
        // TODO implement here
    }

    public void deleteClass() {
        // TODO implement here
    }

}
```

Csv.cs

Лістинг програми:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Csv : Report {

    public Csv() {
    }

    public void createFileCsv() {
        // TODO implement here
    }

}
```

LearningContents.cs

Лістинг програми:

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class LearningContents {

    public LearningContents() {
    }

    private string id;

    public string text;

    public DateTime createdAt;

    public DateTime updatedAt;

    public string questionId;

}
```

Material.cs

Лістинг програми:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Material {

    public Material() {
    }

    private string materialId;

    private string subjectId;

    public string title;

    public string description;

    public string precondition;

    public string purpose;

    public DateTime createdAt;

    public DateTime updatedAt;

    public string coverPictureUrl;

    public MaterialStatus status;

    private string ownerId;

    public void getMaterial() {
        // TODO implement here
    }

    public void createMaterial() {
        // TODO implement here
```

```
    }

    public void editMaterial() {
        // TODO implement here
    }

    public void deleteMaterial() {
        // TODO implement here
    }

}
```

MaterialStatus.cs

Лістинг програми:

```
public enum MaterialStatus {
    New,
    InProgress,
    Passed,
    Failed
}
```

Message.cs

Лістинг програми:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Message {

    public Message() {
    }

    private string messageId;

    public string content;

    public string userId;

    public bool isRead;

    public string assignedTo;

    public DateTime createdAt;

    public void markRead() {
        // TODO implement here
    }

    public void getMessage() {
        // TODO implement here
    }

    /// <summary>
    /// @param Message
    /// </summary>
    public void create(void Message) {
```

```
        // TODO implement here
    }

    public void editMessage() {
        // TODO implement here
    }

    public void deleteMessage() {
        // TODO implement here
    }

}
```

Pdf.cs

Лістинг програми:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Pdf : Report {

    public Pdf() {
    }

    public void createFilePdf() {
        // TODO implement here
    }

}
```

Progress.cs

Лістинг програми:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Progress {

    public Progress() {
    }

    public string progressId;

    public string userId;

    public string materialId;

    public int statusId;

}
```

Pupil.cs

Лістинг програми:

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;

public class Pupil : User {

    public Pupil() {
    }

    public void idClass;

}
```

PupilReport.cs

Лістинг програми:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class PupilReport : Report {

    public PupilReport() {
    }

    public void getReports() {
        // TODO implement here
    }

}
```

Question.cs

Лістинг програми:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Question {

    public Question() {
    }

    private string id;

    public string title;

    public DateTime createdAt;

    public DateTime updatedAt;

    public string feedbackCorrectText;

    public string feedbackIncorrectText;

    public string materialId;

    public void getQuestion() {
        // TODO implement here
    }
```

```
    public void createQuestion() {
        // TODO implement here
    }

    public void editQuestion() {
        // TODO implement here
    }

    public void deleteQuestion() {
        // TODO implement here
    }

}
```

RankingTextAnswers.cs

Лістинг програми:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class RankingTextAnswers {

    public RankingTextAnswers() {
    }

    private string id;

    public string text;

    public DateTime createdAt;

    public DateTime updatedAt;

    public string questionId;

}
```

Report.cs

Лістинг програми:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public interface Report {

    public void getReports();

    public void parseData();

    public void extractData();

    public void createFile();

}
```

SingleSelectImageAnswers.cs

Лістинг програми:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class SingleSelectImageAnswers {

    public SingleSelectImageAnswers() {
    }

    private string id;

    public string image;

    public DateTime createdAt;

    public DateTime updatedAt;

    public bool isCorrect;

    public string questionId;

}
```

Subject.cs

Лістинг програми:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Subject {

    public Subject() {
    }

    private string subjectId;

    public string subjectName;

    private string ownerId;

    public DateTime createdAt;

    public DateTime updatedAt;

    public string classId;

    public void getSubject() {
        // TODO implement here
    }

    public void createSubject() {
        // TODO implement here
    }

    public void editSubject() {
        // TODO implement here
    }
```

```
    public void deleteSubject() {
        // TODO implement here
    }

}
```

Teaher.cs

Лістинг програми:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Teacher : User {

    public Teacher() {
    }

    public Array<string> speciality;

    public void manageSubject() {
        // TODO implement here
    }

    public void manageMaterial() {
        // TODO implement here
    }

}
```

TextMatchingAnswers.cs

Лістинг програми:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class TextMatchingAnswers : Question {

    public TextMatchingAnswers() {
    }

    private string id;

    public string value;

    public DateTime createdAt;

    public DateTime updatedAt;

    public string questionId;

}
```

User.cs

Лістинг програми:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class User {

    public User() {
    }

    private string userId;

    public string email;

    public string fullName;

    public Date birthDate;

    public Date joinDate;

    public string avatarUrl;

    protected varchar passwordHash;

    protected varchar passwordResetToken;

    public void addMessage() {
        // TODO implement here
    }

    public void deleteMessage() {
        // TODO implement here
    }

}
```

Xls.cs

Лістинг програми:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Xls : Report {

    public Xls() {
    }

    public void createFileXls() {
        // TODO implement here
    }

}
```