Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для закладів дошкільної освіти з керуванням режимом дня та харчуванням дітей. Back-end
(тема)

Виконав:
студент 4 курсу, групи ПЗПІ-20-1

Смєйко Б.М.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. Побіженко І.О.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Смєйку Богдану Миколайовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Програмна система для закладів дошкільної освіти з керуванням режимом дня та харчуванням дітей. Back-end
Затверджена наказом по університету від 20.05.2024р. № 471 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 14.06.2024
3. Вихідні дані до роботи Розробити серверну частину програмної системи для закладів дошкільної освіти з керуванням режимом дня та харчуванням дітей, що надаватиме необхідний функціонал батькам, вихователям та адміністраторам закладу, використовуючи мову програмування C# та сучасні технології.
4. Перелік питань, що потрібно опрацювати в роботі
Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	08.04.2024	*виконано*
2	Створення специфікації ПЗ	12.04.2024	*виконано*
3	Проектування ПЗ	15.04.2024	*виконано*
4	Розробка ПЗ	15.05.2024	*виконано*
5	Тестування ПЗ	17.05.2024	*виконано*
6	Оформлення пояснювальної записки	01.06.2024	*виконано*
7	Підготовка презентації та доповіді	06.06.2024	*виконано*
8	Попередній захист	16.06.2024	*виконано*
9	Нормоконтроль, рецензування	16.06.2024	*виконано*
10	Здача роботи у електронний архів	16.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	17.06.2024	*виконано*

Дата видачі завдання 8 квітня 2024р.

Студент (ка)_____ Смєйко Б.М.
(підпис)

Керівник роботи _____ доц. Побіженко І.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 115 стор., 40 рис., 2 табл., 15 джерел.

АВТОМАТИЗАЦІЯ, БЕЗПЕКА, ДИТЯЧІ САДКИ, РОЗКЛАД ДНЯ, ШТУЧНИЙ ІНТЕЛЕКТ, OAUTH2

Об’єкт розробки – програмна система для закладів дошкільної освіти з керуванням режимом дня та харчуванням дітей. Back-end.

Мета розробки – створення серверної частини програмної системи для закладів дошкільної освіти з керуванням режимом дня та харчуванням дітей, що надаватиме функціонал батькам, вихователям та адміністраторам закладу.

Метод рішення – хмарні платформи Azure та Google Cloud Platform (GCP), мова програмування C# та середовище розробки JetBrains Rider.

У результаті розроблено серверну частину програмної системи для закладів дошкільної освіти з керуванням режимом дня та харчуванням дітей, що надає функціонал батькам, вихователям та адміністраторам закладу.

AUTOMATION, SECURITY, PRESCHOOLS, SCHEDULE OF THE DAY, ARTIFICIAL INTELLIGENCE, OAUTH2

The object of development is a back-end part of the software system for preschool education institutions with management of the daily schedule and nutrition of children.

The purpose of the work is to create back-end part of the software system for preschool education institutions with management of the daily schedule and nutrition of children, which will provide functionality to parents, teachers and administrators of the institution.

Solution method – cloud platforms such as Azure and Google Cloud Platform (GCP), C# programming language and JetBrains Rider development environment.

As a result, back-end part of the software system for preschool education institutions with management of the daily schedule and nutrition of children was developed, which provides functionality to parents, teachers and administrators of the institution.

Я, Смєйко Богдан Миколайович, студент гр. ПЗПІ-20-1, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система для закладів дошкільної освіти з керуванням режимом дня та харчуванням дітей. Back-end», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

Сучасний світ характеризується стрімким розвитком інформаційних технологій, що впливає на всі сфери життя, в тому числі й на систему дошкільної освіти. Дитячі садки, як важливий елемент освітнього процесу, також потребують інтеграції сучасних технологій для підвищення ефективності роботи та покращення комунікації з батьками. Проте, багато з цих закладів ще не здійснили перехід до сучасних інформаційних технологій в управлінні своєю діяльністю. Більшість з них використовують застарілі методи та ручні процеси для ведення обліку дітей, розкладу дня, харчування, та взаємодії з батьками. Крім цього багато дошкільних закладів ще використовують паперову документацію для ведення обліку відвідування дітьми садка, проведення медичних процедур, та інших аспектів діяльності. Це призводить до великої кількості ручної роботи для вихователів та адміністраторів, а також збільшує ризик втрати чи неправильного обліку даних. Більшість дошкільних закладів не мають централізованої системи управління, що ускладнює координацію роботи між вихователями, адміністрацією та батьками. Інформація про режим дня, харчування, та успішність дітей часто розподілена по різних джерелах, що може призводити до неоднозначності та плутанини. Батьки ж в свою чергу, зазвичай, не мають зручного доступу до інформації про свою дитину під час перебування в дошкільному закладі. Інформація про режим дня, харчування, та активності дітей не завжди доступна в реальному часі, що ускладнює взаємодію між закладом та батьками.

Метою даної кваліфікаційної роботи є створення серверної частини комплексної програмної системи для дитячих садків, яка має на меті цифровізацію процесів закладу та забезпечення зручного доступу до інформації для батьків, вихователів та адміністраторів закладів. Система буде базуватися на використанні сучасних технологій, включаючи штучний інтелект (ШІ) та хмарні обчислення. Крім цього, за допомогою тимчасових QR-кодів та двофакторної автентифікації, система матиме підвищену систему захисту та доступу до

інформації, що дуже важливо забезпечити, розробляючи сучасні додатки. Основним завданням цієї роботи є створення функціональності програмної системи для подачі заявок в дитячі садки, автоматизованої генерації розкладу з використанням ШІ, системи відвідування, системи рахунків за сплату користування послугами садків та нагадуваннями про терміни сплати, а також розробка серверу автентифікації (Identity Server) для реалізації єдиної системи входу (SSO), що дозволить користувачам входити до різних додатків і сервісів системи, використовуючи одні й ті ж облікові дані без необхідності повторного введення логіну та пароля. Крім того, це дозволить ефективно і безпечно управляти доступом до ресурсів та додатків, зменшуючи ризик витоку даних та спрощуючи адміністрування облікових записів користувачів. У результаті SSO забезпечить зручність для користувачів і підвищить продуктивність, а також сприятиме підвищенню безпеки та контролю за доступом до системи.

Результати даної роботи будуть мати практичне застосування у дошкільних освітніх закладах, дозволяючи їм ефективно управляти як освітніми так і організаційними процесами, а для батьків можливість бути максимальному наближеними до усіх процесів пов’язаних з їхніми дітьми в садках, мати прозору, детальну, а головне в реальному часі інформацію. Це сприятиме зменшенню відстані між садочком та батьками та їх залученості до внутрішніх процесів. Крім того, розроблена програмна система може стати основою для подальшого впровадження сучасних цифрових технологій у сфері дошкільної освіти.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Сучасний ринок програмного забезпечення для дитячих садків демонструє зростаючий інтерес до цифровізації процесів. Це обумовлено декількома чинниками:

- зростання потреби в прозорості та доступі до інформації: батьки все більше цікавляться життям своїх дітей в садочку, а вихователі та адміністратори шукають ефективні інструменти для спілкування з батьками та обміну інформацією;
- потреба в автоматизації рутинних процесів: керування режимом дня, харчуванням, відвідуванням та іншими аспектами роботи садочка займає значний час у вихователів та адміністраторів. Автоматизація цих процесів дозволяє звільнити час для більш важливих завдань;
- зростання популярності мобільних технологій: сучасні батьки та вихователі активно користуються смартфонами та планшетами, тому важливо, щоб програмне забезпечення було доступне з будь-якого пристрою та в будь-який час.

Відтак, це приводить нас до необхідності докладного вивчення слабких місць та проблем, з якими стикаються зацікавлені сторони при використані тих чи інших рішень, які вже є на ринку, та розробки ефективних стратегій їх вирішення. Тільки шляхом їхнього аналізу й пошуку оптимальних рішень вдасться забезпечити успішне впровадження цифрових технологій в управління дитячими садками та максимально задовольнити потреби всіх учасників.

1.2 Виявлення та вирішення проблем

Після детального аналізу ринку було з’ясовано, що існує декілька програмних продуктів для дитячих садків. Серед них виділено 5 систем, що є провідними у даній ніші:

- KidKare [1];

– Brightwheel [2];
– EZChildTrack [3];
– Procare Software (також відомий як Kinderlime) [4];
– Lillio (також відомий як HiMama) [5].

KidKare – це програмна система, розроблена спеціально для дошкільних закладів, яка спрощує керування дитячими записами, відстеження присутності та формування рахунків та загалом має дуже широкий функціонал, призначений для використання адміністративним персоналом дитячих дошкільних закладів (див. рис. 1.1).

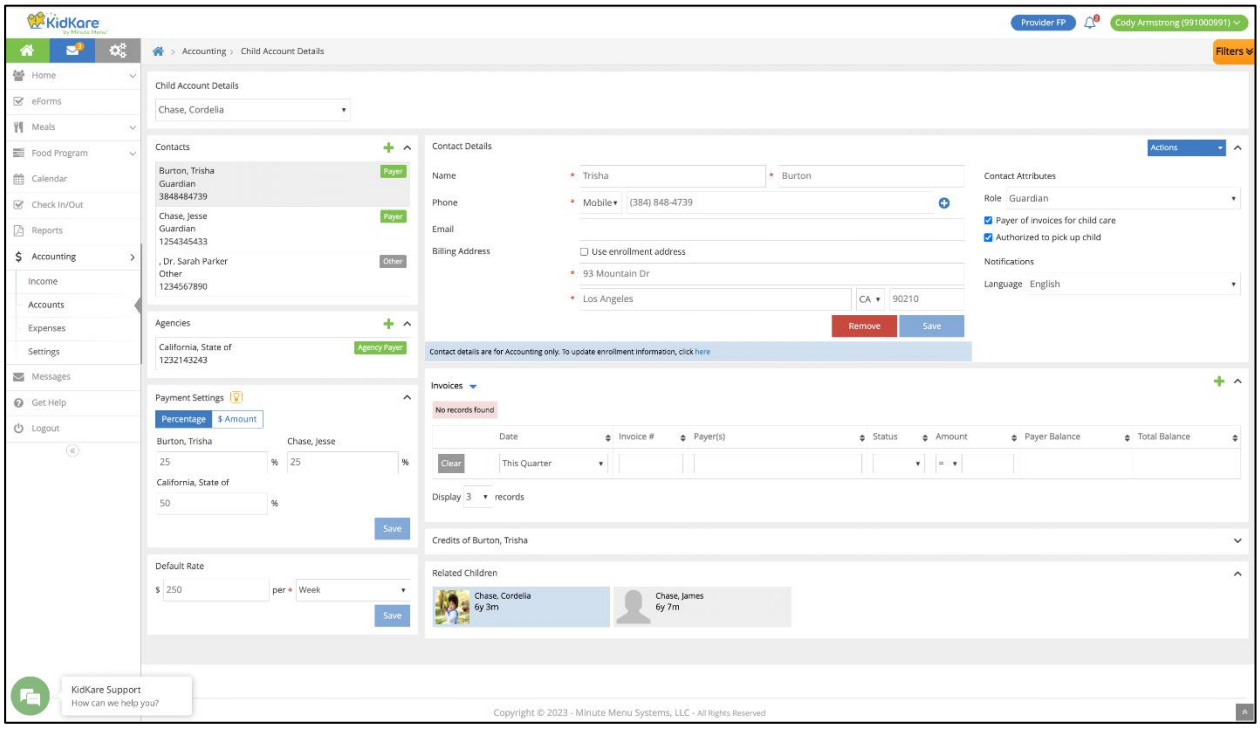


Рисунок 1.1 – Інтерфейс веб-застосунку KidKare (за даними [1])

Серед переваг даного програмного продукту варто виділити:

– реєстрація: адміністратори можуть реєструвати нових дітей у садку;
– керування дитячими записами: зберігання та оновлення даних про дітей, включаючи особисту інформацію, медичні дані, контактну інформацію батьків тощо;
– присутність: система дозволяє відстежувати присутність та відсутність;
– рахунки: застосунок KidKare дозволяє виставляти рахунки для сплати;

– повідомлення: застосунок має нагадування про сплату рахунків.

Окрім переваг, було знайдено і недоліки. Серед них:

– планування режиму дня: система не надає даного функціоналу;
– обмежений функціонал: загалом KidKare призначений для використання адміністраторами закладу, а батьки мають змогу сплачувати рахунки;
– залежність від Інтернет-з'єднання: оскільки KidKare – це хмарний сервіс, він потребує стабільного Інтернет-з'єднання для роботи, що може бути недоліком у випадку проблем з мережею;
– ринок: платформа орієнтована лише на Північну Америку.

Brightwheel – це комплексний інструмент для дошкільних закладів, призначений для спрощення керування і взаємодії між персоналом, батьками та дітьми. Цей застосунок включає в себе можливості планування режиму дня, формування рахунків та відстеження присутності дітей (див. рис. 1.2).

Перевагами даного проекту є:

– реєстрація: Brightwheel дозволяє батькам реєструвати своїх дітей у садку;
– рахунки: система дозволяє батькам оплачувати рахунки через мобільний додаток;
– відстеження присутності: система дозволяє вести облік часу присутності дітей;
– QR-коди: система надає можливість використовувати QR-коди для відмічання присутності;
– ринок: окрім Північної Америки, Brightwheel розширює свою присутність в Європі та в інших регіонах.

Недоліками програмному застосунку є:

– відсутність створення власних занять та формування розкладу з ними;
– деякі функції можуть бути обмежені в безкоштовній версії Brightwheel, і для доступу до повного функціоналу може знадобитися підписка;

– для деяких користувачів може виникнути складність у освоєнні системи на початкових етапах через місцями складний інтерфейс.

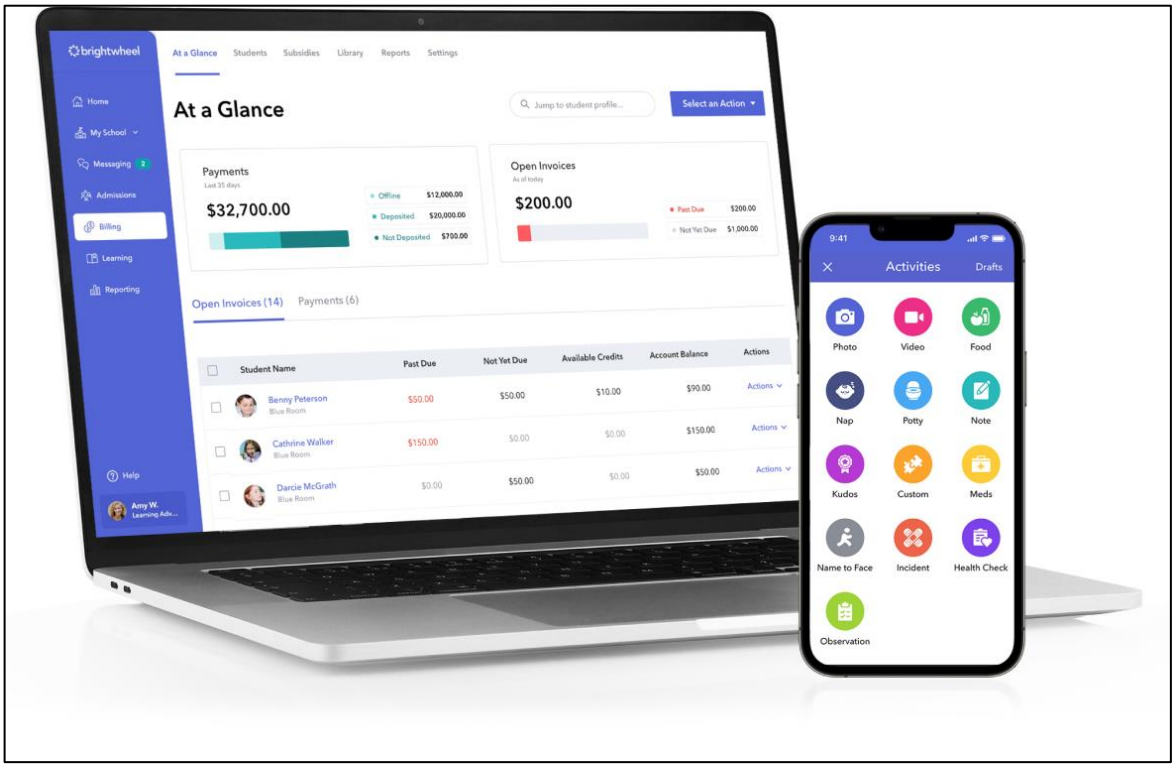


Рисунок 1.2 – Інтерфейси застосунку Brightwheel (за даними [2])

EZChildTrack – веб-орієнтована платформа для керування дитячими центрами та дошкільними закладами. Вона пропонує широкий спектр інструментів для автоматизації процесів в установі, включаючи облік дітей, реєстрацію, планування режиму дня та облік оплати (див. рис. 1.3).

Першочерговими перевагами рішення EZChildTrack є:

– реєстрація та управління: EZChildTrack дозволяє зручно вести реєстрацію нових дітей, керувати даними про них та їхні сім’ї;
– фінансове управління: EZChildTrack надає інструменти для обліку оплати за послуги, створення рахунків, ведення фінансової звітності та роботи з платіжними системами;
– відстеження присутності: система дозволяє вести облік часу присутності дітей;

– QR-коди: система надає можливість використовувати QR-коди для відмічання присутності;
– батьківський портал: EZChildTrack надає функціонал для батьків з можливістю сплати рахунків та редагуванням даних дітей.

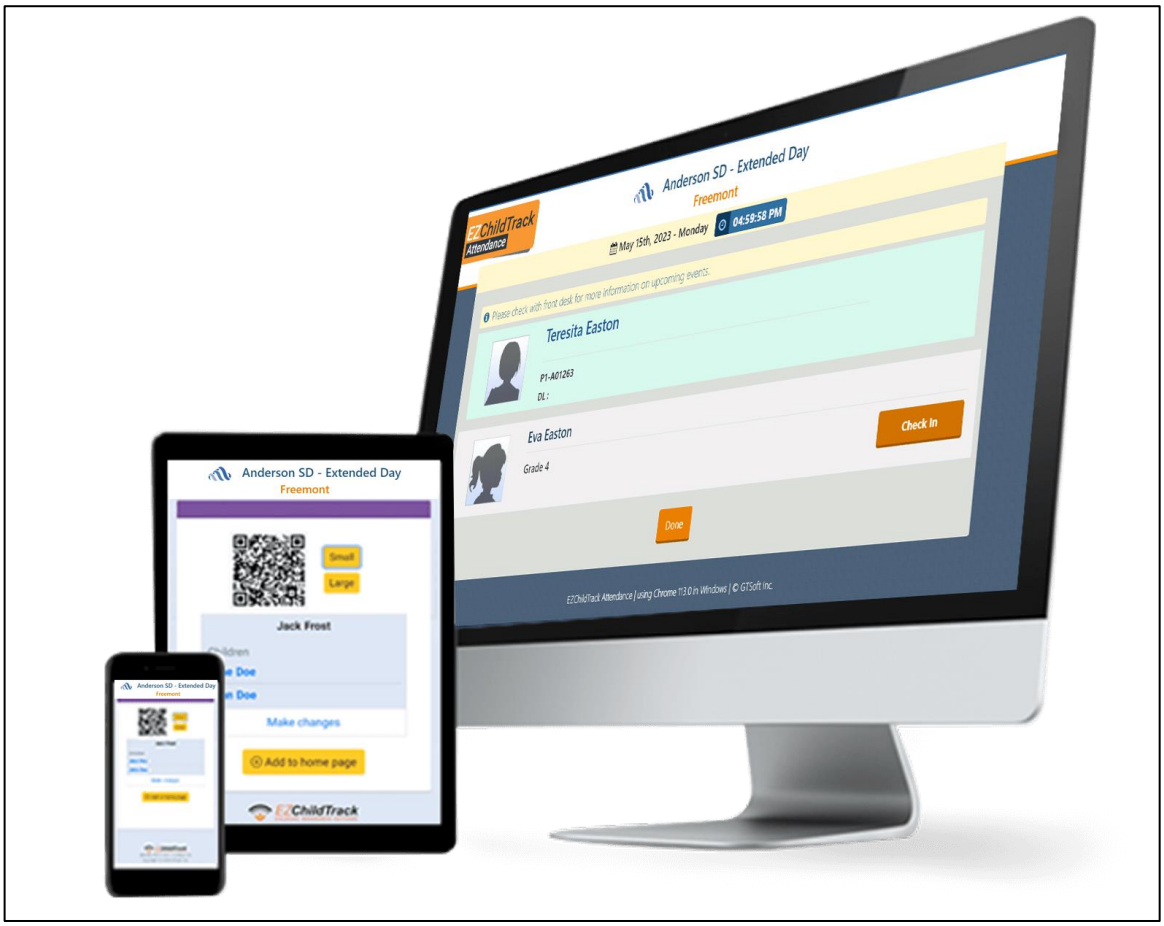


Рисунок 1.3 – Інтерфейси застосунку EZChildTrack (за даними [3])

Недоліки:

– планування режиму дня: система не надає можливості створювати заняття та управляти розкладом дітей в групах;
– складність в освоєнні: деякі користувачі можуть відчувати складність у використанні системи на початкових етапах через обсяг функціоналу;
– ринок: основна спрямованість цієї платформи на ринок Північної Америки, зокрема, у США, де вона активно використовується в дошкільних закладах.

Procare Software (також відомий як Kinderlime) – це програмне рішення, спеціально розроблене для дошкільних закладів, щоб спростити керування дітьми, їхнім розкладом, присутністю та комунікацією з батьками (див. рис. 1.4).

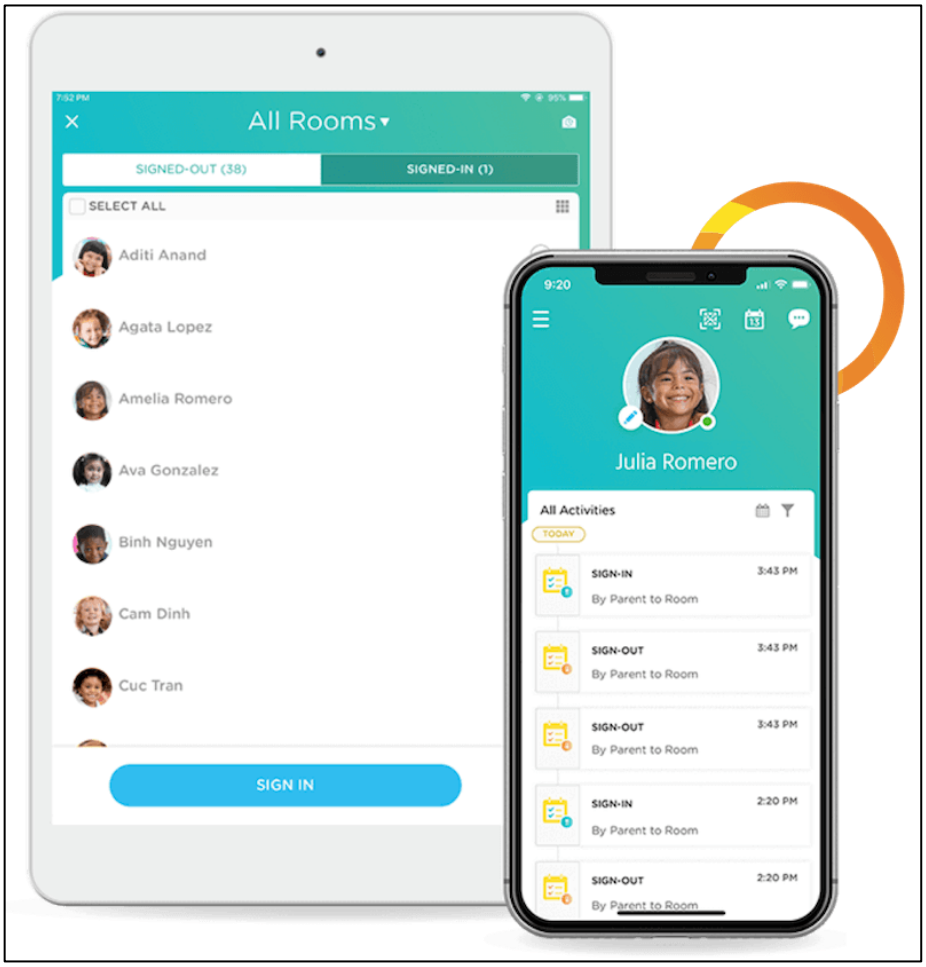


Рисунок 1.4 – Інтерфейси застосунку Procare (за даними [4])

Переваги даного застосунку:

– реєстрація: дозволяє батькам реєструвати та виписувати дитину через мобільний додаток;
– планування режиму дня: система дозволяє встановлювати та відстежувати графік дня, занять та активностей для дітей у дошкільному закладі;
– заняття: застосунок надає можливість створювати власні заняття та формувати з них розклад;

- рахунки: система дозволяє батькам оплачувати рахунки за допомогою мобільного додатка або веб-порталу;
- QR-коди: система надає можливість використовувати QR-коди для відмічання присутності.

Окрім вище зазначених переваг, для Procare характерні недоліки:

- автоматизація: відсутність автоматизованого створення оптимального розкладу;
- специфічність функціональності: деякі користувачі відзначають, що деякі функції можуть бути специфічними або недостатньо гнучкими для їх потреб;
- ринок: платформа зосереджена лише на Північній Америці та поширює свою присутність на Латинську Америку.

Lillio (також відомий як HiMama) – це програмна система, спрямована на полегшення комунікації між вихователями та батьками у дитячих садках (див. рис. 1.5).

Серед переваг системи варто виділити наступні:

- реєстрація: дозволяє батькам реєструвати своїх дітей у садку;
- розклад: вихователі можуть створювати календарні події, такі як свята, важливі дати чи події в закладі;
- присутність: система дозволяє відстежувати присутність та відсутність;
- рахунки: система дозволяє батькам сплачувати рахунки;
- ринок: платформа використовується на ринках Північної Америки та Європи.

Для Lillio характерні наступні недоліки:

- QR-коди: система не надає можливості використовувати QR-коди для відмічання присутності;
- планування режиму дня: система не надає можливості встановлювати та відстежувати графік дня, занять та активностей для дітей;

– складність в освоєнні інтерфейсу: для деяких користувачів інтерфейс Lillio може виявитися менш інтуїтивно зрозумілим або складнішим у використанні порівняно з іншими аналогічними програмами;
– обмежений функціонал: загалом Lillio призначений для використання батьками та вихователями, без адміністраторів закладу.

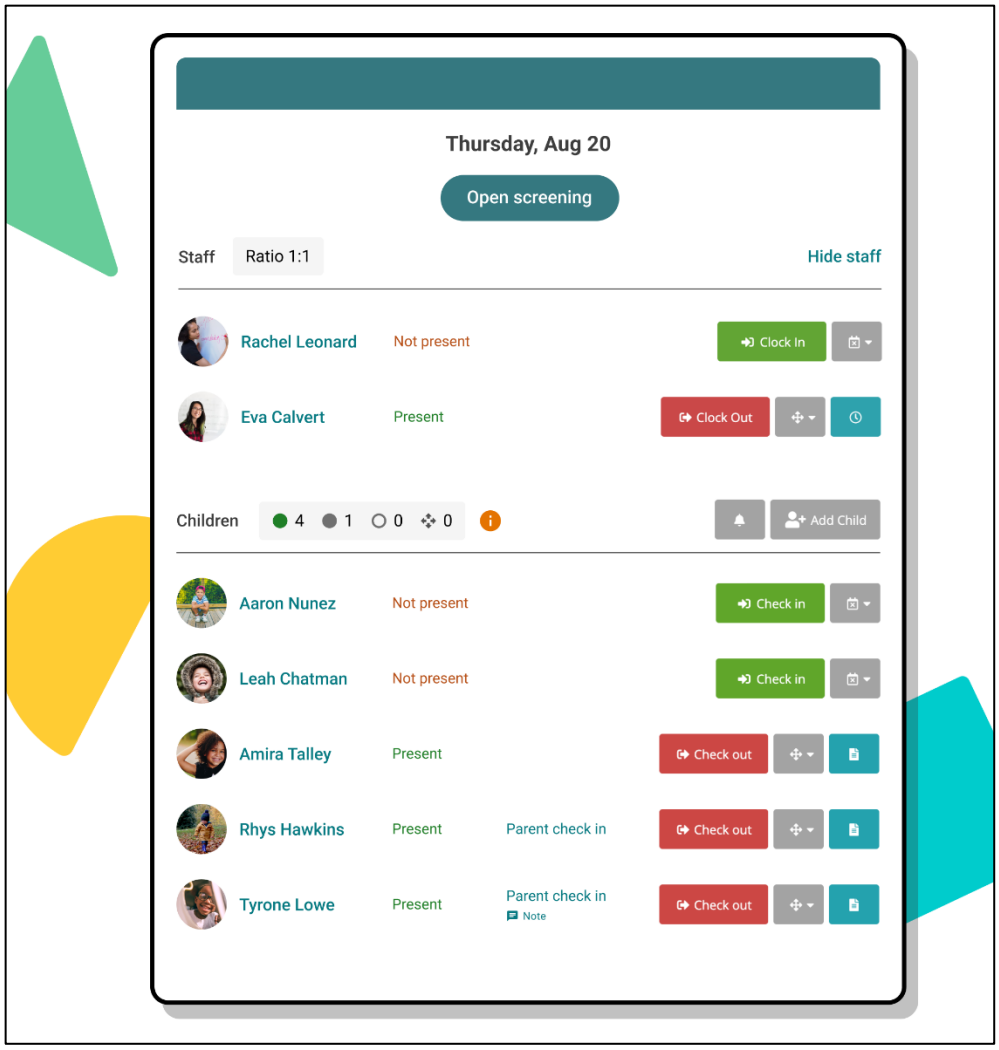


Рисунок 1.5 – Інтерфейс застосунку Lillio (за даними [5])

Отже, на ринку представлені різні програмні продукти для дитячих садків, що мають різні функціональні можливості, але багато з цих систем мають обмежений функціонал та зосереджені на окремих аспектах роботи садка, а не на комплексному управлінні, що обмежує їх використання для батьків, вихователів або адміністраторів закладу. Відсутність універсальної системи

ускладнює взаємодію між батьками, вихователями та адміністративним персоналом через несумісність інтерфейсів різних програмних систем.

1.3 Постановка задачі

Проаналізувавши функціональні можливості застосунків конкурентів, створено таблицю, що дозволяє їх порівняти, визначити лідера за існуючими можливостями, побачити слабкі сторони, місця для покращення та визначити необхідні функції для впровадження в розроблюваному додатку. Порівняння відбувалося за такими критеріями:

– зручна реєстрація дітей та подання заяв до садків (А);

– відстеження присутності та QR-коди (Б);

– планування режиму дня (В);

– створення власних занять (Г);

– автоматичне складання/розподіл розпорядку дня (Д);

– рахунки (Е);

– адаптованість до ринку України (Є);

– функціонал для батьків (Ж);

– функціонал для вихователів (З);

– функціонал для адміністраторів (Д).

Наочно наявність різного функціоналу відображено у таблиці 1.1 для кожного із програмних продуктів, що розглядаються.

З огляду на відсутність певного функціоналу на ринку в програмному забезпеченні конкурентів і на відмінність цього функціоналу, його розробка і впровадження набуває критичного значення. Ця необхідність пояснюється тим, що існуючі програми не можуть повністю задовольнити потреби користувачів через відсутність специфічних функцій, які можуть бути критично важливими для їхніх потреб. Відтак, створюється великий потенціал зайняти цей нішевий ринок і залучити до свого продукту широке коло користувачів, надаючи їм

доступ до раніше недоступних можливостей, що здатне значно підвищити їхній досвід використання подібних систем.

Таблиця 1.1 – Порівняння існуючих рішень на ринку

Програмний продукт	А	Б	В	Г	Д	Е	Є	Ж	З	Д
KidKare	+/-	+	-	-	-	+	-	+\-	+	+
Brightwheel	+	+	+	-	-	+	+/-	+	+	+
EZChildTrack	+	+	+	-	-	+	-	+	+/-	+
Procare	+	+	+	+	-	+	-	+	+	+
Lillio	+	+/-	-	-	-	+	+/-	+	+	-
Програмний продукт, що розробляється	+	+	+	+	+	+	+	+	+	+

Отже, необхідно розробити програмну систему для закладів дошкільної освіти, яка б:

- включала в себе весь необхідний функціонал, визначений в таблиці 1.1, а саме: зручну реєстрацію дітей та подання заяв до садків (А), відстеження присутності та QR-коди (Б), планування режиму дня (В), створення власних занять (Г), автоматичне складання/розподіл розпорядку дня з використанням ШІ (Д), рахунки (Е), адаптованість до ринку України (Є), функціонал для батьків (Ж), функціонал для вихователів (З), функціонал для адміністраторів (Д);
- заповнила прогалини на ринку, реалізувавши унікальний функціонал, який відсутній у конкурентів і є критично важливим для користувачів;
- забезпечила конкурентну перевагу, надаючи користувачам доступ до нових, раніше недоступних можливостей;
- задовольнила потреби широкого кола користувачів, включаючи батьків, вихователів та адміністраторів дитячих садків;

– сприяла покращенню досвіду користувачів під час використання систем управління дитячим садком.

Розробка та впровадження відсутнього функціоналу є критично важливим для успіху програмної системи. Це дозволить залучити широке коло користувачів та зайняти нішу на ринку, надаючи користувачам доступ до унікальних можливостей.

1.3.1 Цільова аудиторія

Програмна система націлена вирішити проблему залученості усіх сторін дитячих садків, тому передбачається, що цільовою аудиторією програмної системи будуть:

а) адміністративний персонал закладу, який прагне:
   1) підвищення ефективності роботи закладу та зменшення адміністративного навантаження;
   2) автоматизації рутинних завдань, отримання своєчасної та достовірної інформації, прозорість управлінських рішень.

б) вихователі закладу, які прагнуть:
   1) підвищення якості освітнього процесу та полегшення взаємодії з батьками;
   2) зменшення паперової роботи, більше часу для роботи з дітьми, прозора взаємодія з батьками.

в) батьки дітей, які прагнуть:
   1) отримання повної та достовірної інформації про перебування дитини в дитячому садку;
   2) бути в курсі всіх подій, мати можливість оперативно зв’язатися з вихователями, зручно та швидко оплачувати послуги.

Тобто програмна система має на меті охопити якомога більшу аудиторію, включаючи їхні різні кінцеві цілі у використанні системи та підходить для повноцінного використання без залучення додаткових програмних засобів.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Визначення вимог є критично важливим для успішної подальшої роботи над системою. Вимоги визначають функціональні та нефункціональні характеристики системи, які необхідно задовольнити, щоб вона відповідала потребам користувачів і була ефективною у використанні.

Для забезпечення гнучкості розробки та модульності системи, серверна частина складатиметься з трьох частин (сервісів):

- основний API;
- сервер автентифікації;
- адміністрування сервера автентифікації.

Основний API системи буде розроблено за принципами REST та повинен взаємодіяти з клієнтськими додатками за HTTPs протоколом. Мета цього API це надання основного функціоналу системи для батьків, вихователів та адміністраторів дитячих садків.

Сервер автентифікації відповідатиме за управління користувачами та їх реєстрацію та буде реалізований за протоколом OAuth2, який має на меті дозволити доступ із клієнтської програми (веб-сайту, мобільного застосунку, будь-якого іншого пристрою, підключеного до Інтернету тощо) до захищеного ресурсу (основного API), від імені власника ресурсу (кінцевого користувача). Цей протокол може взаємодіяти з різними транспортними протоколами, але він дуже популярний для захисту веб-сервісів REST [6].

Адміністрування сервера автентифікації в свою чергу дозволить керувати усіма користувачами системи, ролями, API ресурсами та областями, а також надасть зручний інтерфейс управління та налаштування програмних клієнтів, таких як веб, мобільні та десктопні додатки, або інші сервери та мікросервіси, що повинні мати змогу використовувати розроблюваний сервер для своєї автентифікації.

2.1 Функціональні вимоги

Нижче наведено функціональні вимоги основного програмного інтерфейсу (API) системи:

- додавання дітей;
- подання заявок до дитячих садків;
- прийняття або відхилення заявок батьками та адміністраторами;
- додавання дітей в групи;
- створення занять та їх налаштування;
- генерація розкладу за допомогою штучного інтелекту;
- експорт розкладу до Google-календаря;
- генерація тимчасових QR-кодів для дітей;
- відмічання присутності дитини за допомогою QR-кодів;
- автоматичне створення рахунків;
- сплата рахунків;
- повідомлення на пошту щодо створених та сплачених рахунків.

Нижче наведено функціональні вимоги для сервера автентифікації, що повинен бути розроблений за протоколом OAuth2:

- реєстрація та автентифікація користувачів;
- реєстрація та автентифікація користувачів за допомогою Google;
- можливість застосування двофакторної автентифікації;
- редагування даних профілю;
- зміна або скидання паролю.

Далі наведено функціональні вимоги для адміністрування сервера автентифікації:

- управління ролями системи;
- управління користувачами системи;
- управління програмними клієнтами системи.

2.2 Нефункціональні вимоги

Нижче наведено функціональні вимоги до усіх сервісів серверної частини, включаючи основний API, сервер автентифікації та адміністрування:

- система повинна підтримувати інтернаціоналізацію та локалізацію для української та англійської мови;
- система повинна впроваджувати контроль доступу на основі ролей, щоб гарантувати, що користувачі мають доступ лише до ресурсів і функцій, які відповідають їхнім ролям;
- швидкість відповіді на запит не повинна перевищувати 2 секунди, не враховуючи запити на генерацію розкладу;
- система має бути доступна 99.9% часу, бути надійною та відмовостійкою;
- система повинна легко розширюватися для впровадження нового функціоналу або інтеграції з іншими системами;
- система повинна впроваджувати надійні механізми обробки помилок, що можуть призвести до відмов у роботі системи.

Шляхом впровадження надійних механізмів автентифікації, гнучкої серверної архітектури та розширених можливостей управління, система має бути спрямована на задоволення потреб різних користувачів, включаючи батьків, вихователів та адміністраторів. Функціональність основного API та сервера автентифікації, а також адміністрування, повинні бути ретельно спроектовані для забезпечення ефективної роботи системи, враховуючи найважливіші завдання, що стоять перед дитячими садками та батьками. Крім того, нефункціональні вимоги, такі як безпека, продуктивність, доступність та легкість розширення, повинні гарантувати стабільну та надійну роботу системи навіть в умовах постійних змін та вимог користувачів.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

Перед початком розробки програмної системи було виявлено основні ролі користувачів. В нашому випадку існує 5 видів користувачів:

- неавтентифікований користувач;
- користувач з роллю «Батьки»;
- користувач з роллю «Вихователь»;
- користувач з роллю «Адміністратор закладу»;
- глобальний адміністратор програмної системи.

Для кожного з цих типів користувачів було розроблено Use Case діаграму, яка наочно відображає можливості кожної ролі. На рисунку 3.1 наведено Use Case діаграму для неавтентифікованого користувача.

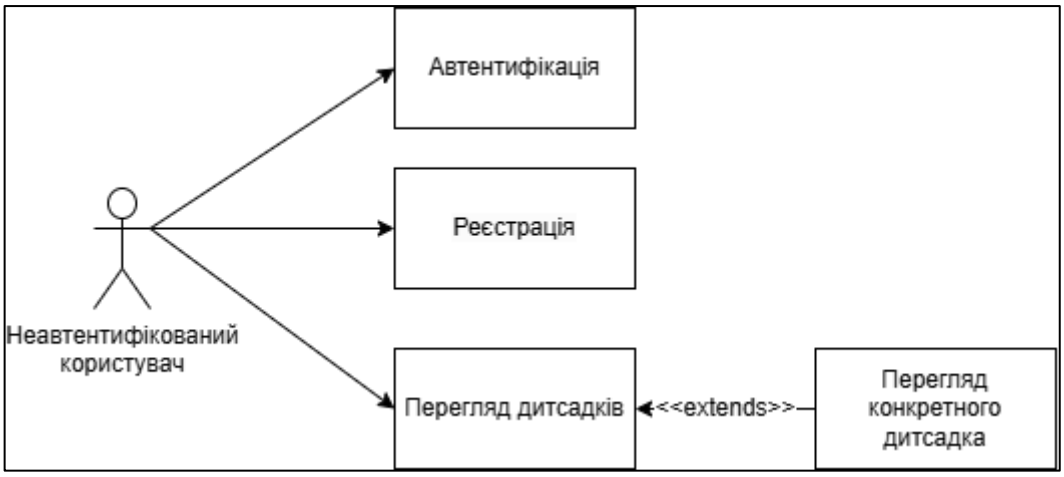


Рисунок 3.1 – Use Case діаграма неавтентифікованого користувача (рисунок виконаний самостійно)

Як видно з діаграми, неавтентифіковий користувач може лише переглядати дитячі садки, що публічно доступні. Крім цього він може зареєструватися або увійти до системи, якщо в нього вже є акаунт.

На рисунку 3.2 наведено Use Case діаграму для користувача з роллю «Батьки».

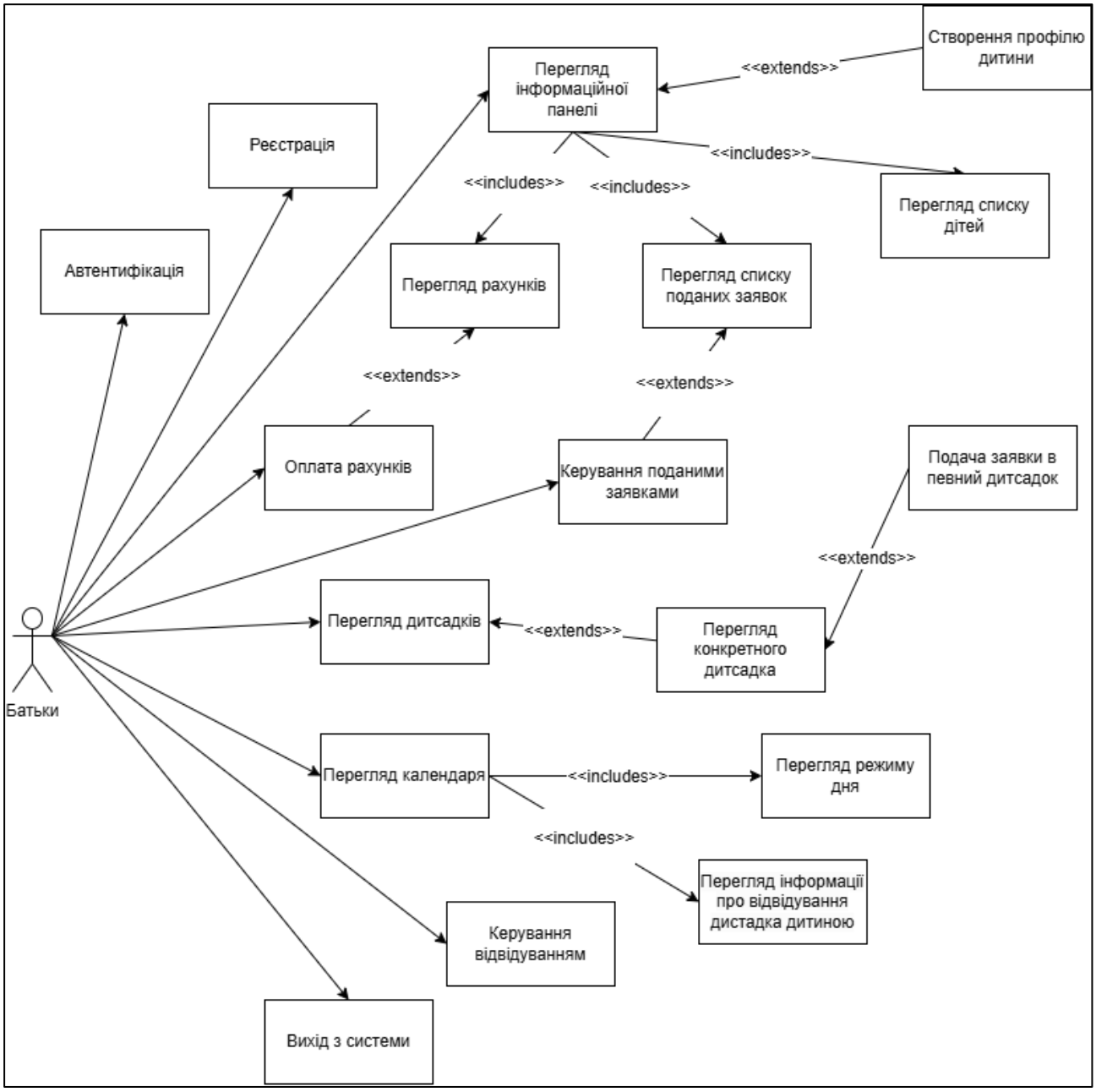


Рисунок 3.2 – Use Case діаграма користувача з роллю «Батьки» (рисунок виконаний самостійно)

Користувачі з роллю «Батьки» можуть реєструватися у системі та користуватися нею як будь-якою іншою системою, до якої вони звикли. Крім цього їм доступно безліч інших функцій, таких як управління профілем дітей, заявками до дитячих садків, сплатою рахунків, а також можливість переглядати календар з розкладом дня та інформацією про відвідуваність дитиною садка.

На рисунку 3.3 наведено Use Case діаграму для користувача з роллю «Вихователь».

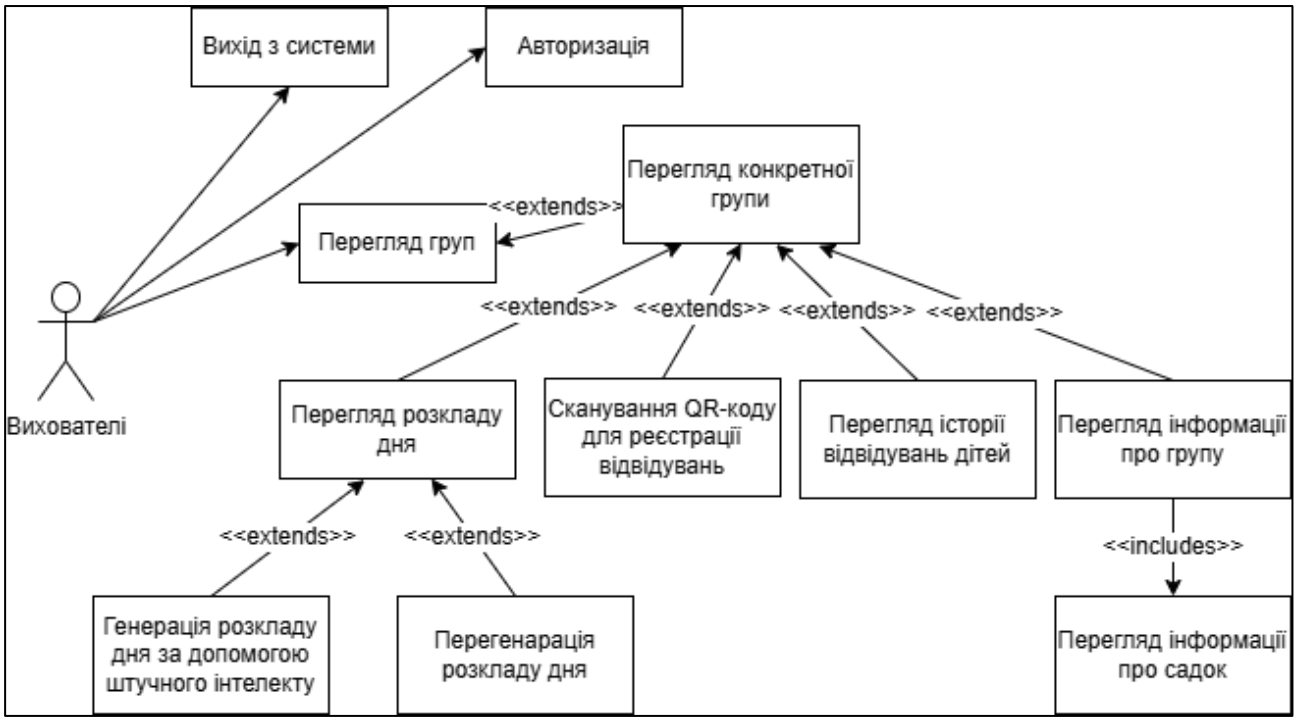


Рисунок 3.3 – Use Case діаграма користувача з роллю «Вихователь» (рисунок виконаний самостійно)

Користувачі з роллю «Вихователь» не можуть самостійно реєструватися в системі, це робить адміністратор закладу, який створює акаунти вихователям, а вони в свою чергу можуть використовувати ці облікові дані для користування системою. Зроблено це з метою забезпечення більшого контролю та безпеки в процесі реєстрації користувачів у системі. Дозволяючи лише адміністраторам закладу створювати облікові записи для вихователів, ми забезпечуємо, що лише авторизований персонал має доступ до системи. Це дозволяє уникнути можливих проблем з некоректною аутентифікацією або недостовірними обліковими записами. Крім того, це забезпечує централізований підхід до управління обліковими даними закладу та спрощує процес створення акаунтів для вихователів, зменшуючи можливість помилок у цьому процесі. Основний функціонал вихователів дозволяє взаємодіяти з батьками, наприклад сканування

QR-коду для відмічання присутності дитини або генерація розкладу дня за допомогою штучного інтелекту (ШІ).

На рисунку 3.4 наведено Use Case діаграму для користувача з роллю «Адміністратор закладу».

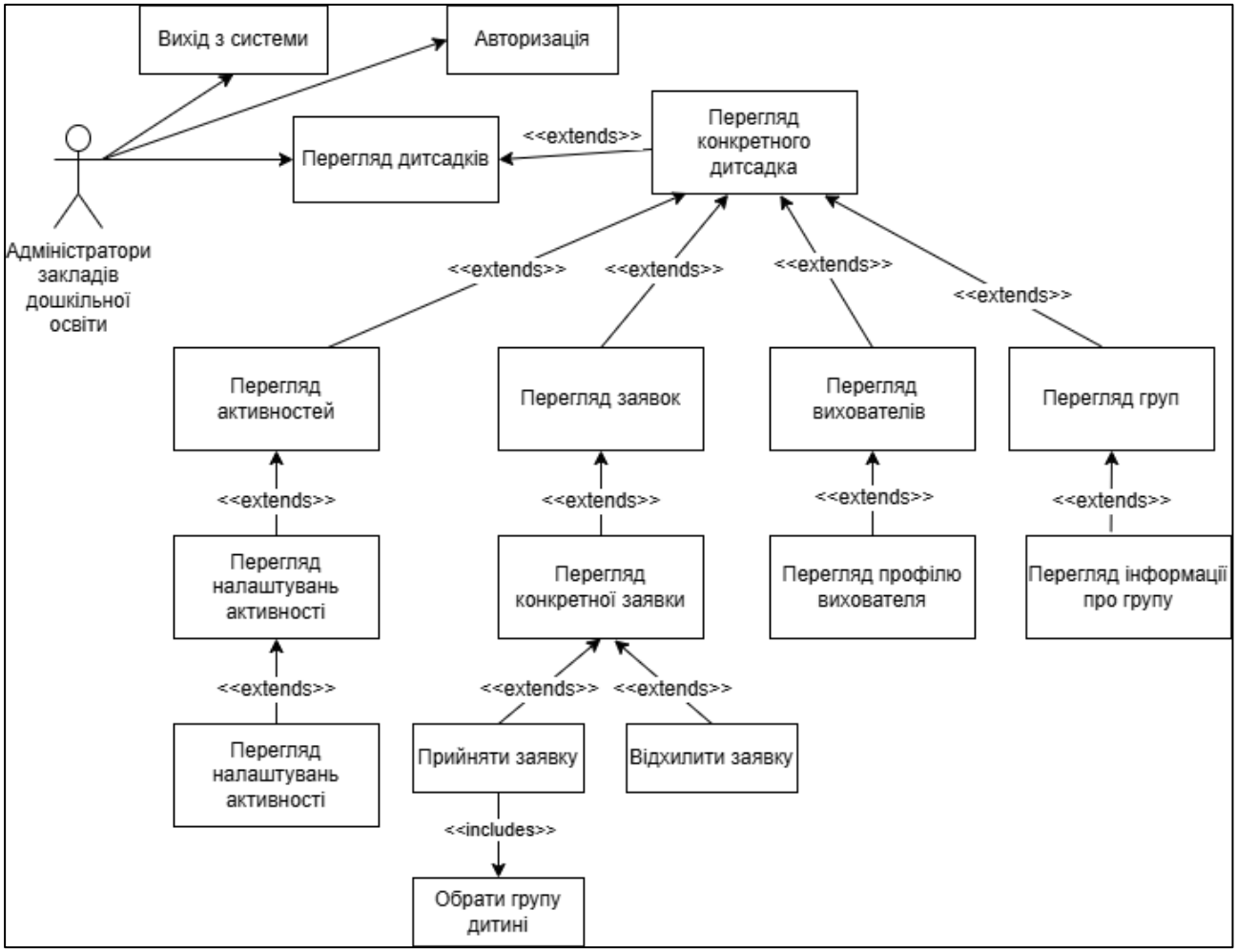


Рисунок 3.4 – Use Case діаграма користувача з роллю «Адміністратор закладу» (рисунок виконаний самостійно)

Користувачі з роллю «Адміністратор закладу» можуть повністю управляти вмістом та налаштуваннями закладу, це включає прийом або відхилення заявок батьків на прийняття їхньої дитини до садка, додавання дитини до групи, створення та налаштування занять (активностей) закладу, які потім використовуються для генерації розкладу дня дітей. А також перегляд усієї супутньої інформації: групи, вихователі та публічні дані закладу.

3.2 Проєктування архітектури ПЗ

3.2.1 REST

Серверну частину програмної системи було реалізовано за архітектурним стилем REST (Representational State Transfer) [7], який визначає набір правил для проектування розподілених гіпермедійних систем. Програмні інтерфейси, що відповідають архітектурному стилю REST, називаються RESTfull або REST API. Принципи, що керують розробкою REST API, значною мірою є результатом вибору архітектури Інтернету, спрямованого на сприяння масштабованості та надійності мережевих ресурсоорієнтованих систем на основі HTTP. До основних принципів REST належить [7, 8]:

- адресність ресурсу: API керують ресурсами, що представляють основні поняття домену, і надають доступ до них; кожен ресурс однозначно ідентифікується та адресується за допомогою відповідного уніфікованого ідентифікатора ресурсу (URI);
- представлення ресурсу: клієнти безпосередньо не знають внутрішнього формату та стану ресурсів; вони працюють із представленнями ресурсів (наприклад, JSON або XML), які представляють поточний або передбачуваний стан ресурсу. Позначення типів вмісту в заголовках повідомлень HTTP дозволяє клієнтам і серверам належним чином обробляти запити;
- уніфікований інтерфейс: доступ до ресурсів та керування ними здійснюється за допомогою стандартних методів, визначених протоколом HTTP (GET, POST, PUT тощо). Кожен метод має власну очікувану, стандартну поведінку та стандартні коди стану;
- відсутність станів: взаємодія між клієнтом і API не має стану, тобто кожен запит містить всю необхідну інформацію для обробки API; стан взаємодії на сервері не зберігається.

Завдяки цьому архітектурному стилю серверна частина надає ресурси, які можна отримувати, створювати, оновлювати та видаляти за допомогою

відповідних HTTP методів, що дозволяє легко інтегрувати систему з іншими додатками.

Розроблений API документується за допомогою Swagger, а у додатку Г наведено OpenAPI специфікацію серверної частини, що містить список усіх реалізованих ендпоінтів, формат їх запитів та відповідей.

3.2.2 CQRS (Command Query Responsibility Segregation)

Також в розробці архітектури було застосовано патерн CQRS, який дозволив розділити операції з доступом до даних на дві окремі категорії: команди (Commands) та запити (Queries) [9]:

- команди (Commands): відповідають за зміну стану системи, наприклад, створення нового об’єкта, оновлення існуючого або його видалення;
- запити (Queries): використовуються для отримання даних з системи без їх зміни.

Розділення команд та запитів забезпечує наступні переваги:

- збільшення продуктивності: команди та запити можуть оптимізуватися окремо, що дозволяє пришвидшити оброблення як запитів, так і команд;
- зменшення складності: кожна частина системи (команди або запити) має чітко визначені завдання, що полегшує розробку, тестування та обслуговування;
- краща масштабованість: команди та запити можуть розподілятися по різних серверах, що дозволяє покращити масштабованість системи.

3.2.3 Mediator

Патерн Mediator відноситься до поведінкових патернів проектування і використовується для спрощення комунікації між компонентами системи, відокремлюючи їх один від одного. У цьому патерні встановлюється спеціальний об’єкт-посередник (Mediator), який координує взаємодію між іншими компонентами. Замість того, щоб об’єкти взаємодіяли напряму, вони спілкуються через цей посередник. Це дозволяє зменшити залежності між

компонентами, підвищує їхню зрозумілість та підтримку, а також спрощує розширення системи [9, 10]. Патерн Mediator часто використовується у великих системах зі складними взаємозв'язками між компонентами, де пряма комунікація може призвести до збільшення залежностей та ускладнення коду.

У нашому проєкті Mediator відповідає за:

- прийом запитів від клієнтів: Mediator приймає HTTP-запити від клієнтів і перенаправляє їх до відповідної команди або запиту;
- виконання команд: Mediator викликає відповідну команду для обробки запиту на зміну стану системи;
- відповідь на запити: Mediator отримує результат виконання команди або запиту і повертає його клієнту.

Використання Mediator спрощує розробку, робить код більш зрозумілим, а також дозволяє легко змінювати систему без необхідності змінювати код інших її частин.

3.2.4 OAuth2

Система реалізує OAuth2 автентифікацію з використанням .NET IdentityServer4, який виступає як централізований сервер авторизації (Authorization Server) для видачі токенів доступу до захищених ресурсів нашого API [10]. Google IdP (скорочено від Google Identity Provider) виступає як зовнішній провайдер ідентифікації, дозволяючи користувачам автентифікуватися за допомогою своїх облікових записів Google.

Різні типи OAuth2 грантів (grant types) визначають, як саме відбувається процес автентифікації. Найпоширенішим є авторизаційний код (Authorization Code Grant), який і було використано в нашій системі. Послідовність дій:

- ініціалізація автентифікації: користувач намагається отримати доступ до захищеного ресурсу;
- перенаправлення на IdentityServer: система перенаправляє користувача на IdentityServer для автентифікації;

- вибір IdP: IdentityServer надає користувачу вибір IdP, включаючи Google. Крім цього користувач може використати локальний акаунт, вже зареєстрований в системі або створити новий;
- перенаправлення на Google IdP: після вибору Google IdP, IdentityServer перенаправляє користувача на сторінку автентифікації Google;
- автентифікація з Google: користувач вводить свої облікові дані Google для автентифікації;
- отримання Authorization Code: після успішної автентифікації, Google IdP перенаправляє користувача назад на IdentityServer з Authorization Code;
- обмін Authorization Code на Access Token: IdentityServer обмінює Authorization Code на Access Token від Google IdP, використовуючи попередньо налаштовані Client ID та Client Secret;
- видача Access Token: IdentityServer видає власний Access Token користувачеві, який містить інформацію про автентифікацію та авторизацію;
- доступ до захищеного ресурсу: користувач може використовувати отриманий Access Token для доступу до захищеного ресурсу.

Використання IdentityServer4 та OAuth2 надає кілька суттєвих переваг:

- централізоване управління автентифікацією: IdentityServer спрощує управління користувачами та їх доступом до ресурсів;
- підтримка зовнішніх IdP: інтеграція з Google IdP дозволяє користувачам використовувати свої існуючі облікові записи Google для автентифікації;
- безпека: OAuth2 забезпечує безпечний спосіб обміну даними авторизації між системою, IdentityServer та Google IdP.

На рисунку 3.5 наведено узагальнену діаграму послідовності (Sequence), на якій наведено усі складові системи, включаючи OAuth сервіс, який забороняє

неавтентифікований доступ до захищених ресурсів нашого API та перенаправляє користувачів на сторінку входу.

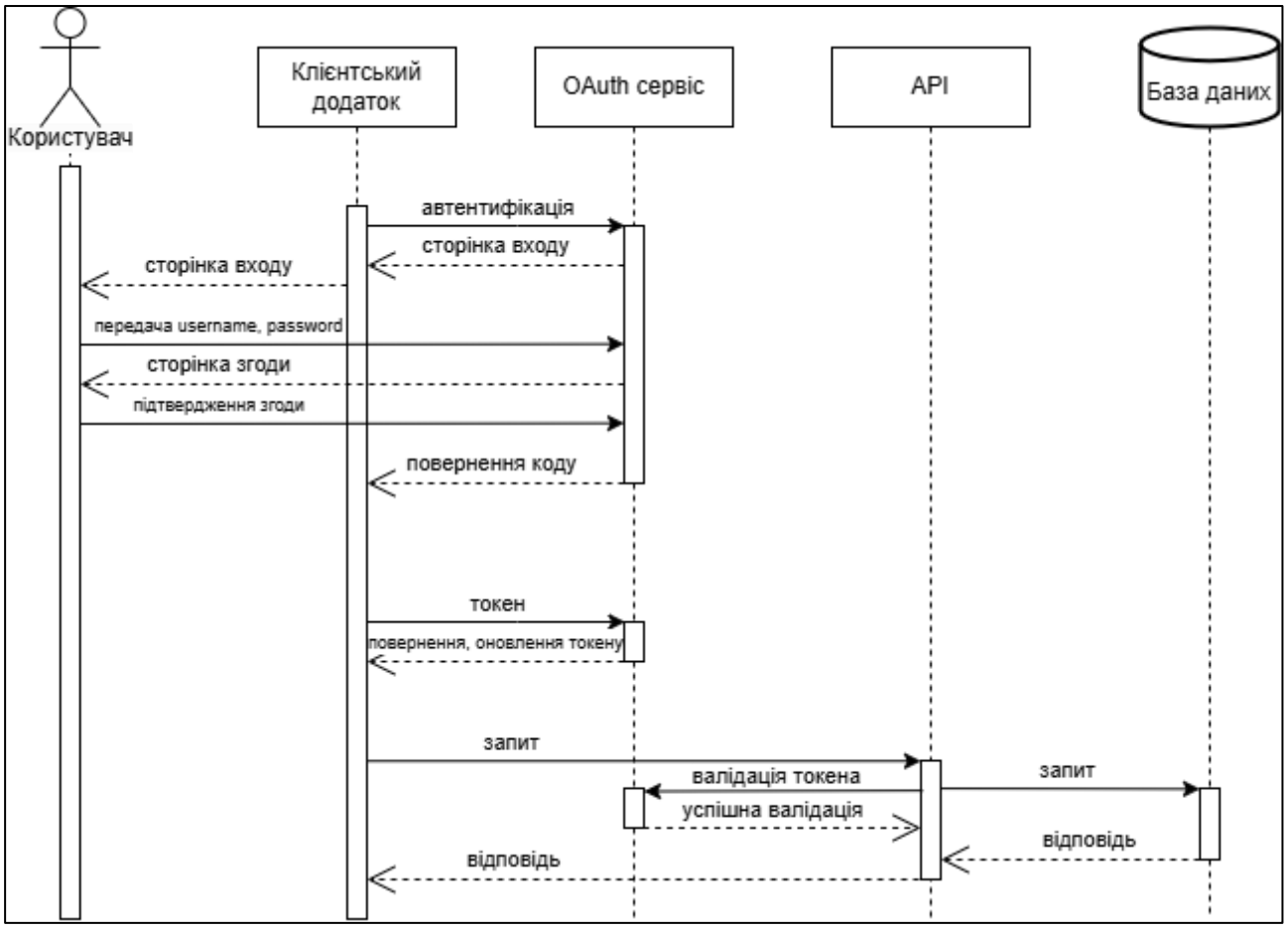


Рисунок 3.5 – Діаграма послідновності (Sequence) програмної системи (рисунок виконаний самостійно)

Загалом, реалізація OAuth2 в нашій системі значно покращує користувацький досвід оскільки він спрощує процес авторизації для користувачів. Замість створення нових облікових записів для кожного додатка, користувачі можуть використовувати свої існуючі облікові записи Google, Facebook, Twitter, замість того, щоб створювати новий обліковий запис.

3.3 Проєктування структури зберігання даних

При проєктуванні структури зберігання даних було розроблено схему бази даних. В нашій системі наявні наступні сутності:

а) «Дитячий садок» (Preschool): основна сутність, яка містить інформацію про дитячий садок, таку як назва, адреса, контактні дані тощо. Зв’язки:
   1) один до багатьох з «Група» (Group): кожен дитячий садок може мати багато груп;
   2) один до багатьох з «Адміністратор дитячого садка» (PreschoolAdmin): кожен дитячий садок може мати багато адміністраторів;
   3) один до багатьох з «Вихователь» (PreschoolTeacher): кожен дитячий садок може мати багато вихователів;

б) «Група» (Group): сутність, що представляє окрему групу в дитячому садку. Зв’язки:
   1) багато до одного з «Дитячий садок» (Preschool): кожна група належить до одного дитячого садка;
   2) один до багатьох з «Дитина» (Child): кожна група може мати багато дітей;
   3) один до багатьох з «Запланована активність» (ScheduleActivity): кожна група може мати багато запланованих активностей;

в) «Дитина» (Child): Сутність, що представляє дитину, яка відвідує дитячий садок. Зв’язки:
   1) багато до одного з «Група» (Group): кожна дитина належить до однієї групи;
   2) один до багатьох з «Заявка» (Application): дитина може бути включеною в багато заявок на вступ до дитячого садка;
   3) один до багатьох з «Відвідуваність» (Attendance): кожна дитина має записи про відвідуваність;

г) «Заявка» (Application): сутність, що містить інформацію про заявку на вступ до дитячого садка. Зв’язки: багато до одного з «Дитина» (Child), де кожна заявка пов’язана з однією дитиною;

д) «Відвідуваність» (Attendance): сутність, що зберігає дані про відвідування дитиною дитячого садка. Зв’язки: багато до одного з

«Дитина» (Child), де кожен запис про відвідуваність пов’язаний з однією дитиною;

е) «Активність» (Activity): сутність, що представляє окрему активність (заняття), яка проводиться в дитячому садку. Зв’язки:
   1) один до багатьох з «Запланована активність» (ScheduleActivity): одна активність може бути запланована багато разів для різних груп;
   2) один до одного з «Налаштування активності» (ActivitySetting): кожна активність може мати одне налаштування;
   3) багато до одного з «Категорія активності» (ActivityCategory): кожна активність належить до однієї категорії;

ж) «Запланована активність» (ScheduleActivity): сутність, що містить інформацію про заплановану активність для конкретної групи. Зв’язки:
   1) багато до одного з «Група» (Group): кожна запланована активність належить до однієї групи;
   2) багато до одного з «Активність» (Activity): кожна запланована активність пов’язана з однією активністю;

з) «Налаштування активності» (ActivitySetting): сутність, що містить додаткові налаштування для конкретної активності. Зв’язки:
   1) один до одного з «Активність» (Activity): налаштування пов’язане з однією активністю;
   2) багато до одного з «Дитячий садок» (Preschool): кожен садок може мати багато налаштувань активностей;

и) «Рахунок» (Invoice): сутність, що містить інформацію про рахунок, виписаний батькам. Зв’язки:
   1) один до багатьох з «Позиція рахунку» (InvoiceItem): кожен рахунок може мати багато позицій;
   2) багато до одного з Дитина (Child): на кожну дитину формуються постійні рахунки.

Ця структура бази даних дозволяє ефективно управляти інформацією про дитячий садок, включаючи дітей, вихователів, активності, відвідуваність, рахунки тощо. В додатку Д наведено повну Entity-Relationship діаграму.

3.4 Приклади найцікавіших алгоритмів та методів

Нижче наведено функцію (Azure Function), що викликається за розкладом та створює рахунки для батьків за послуги дитячих садків.

```
[Function(nameof(InvoiceCreateFunction))]
public async Task RunAsync(
    [TimerTrigger("0 0 9 1 * *")] TimerInfo timer
)
{
    await _invoiceService.CreateInvoicesForAllChildrenAsync();
}
```

Розклад та планування виклику цієї функції задається CRON-виразом, який є рядком, що описує, коли заплановане завдання повинне бути виконане. Цей вирази складається зазвичай з п’яти полів, але Azure потребує шість для точного планування [11]:

- Секунда: 0-59;
- Хвилина: 0-59;
- Година: 0-23;
- День місяця: 1-31;
- Місяць: 1-12 (або Jan-Dec);
- День тижня: 0-6 (Sun-Sat).

В нашому випадку заданий CRON-вираз «0 0 9 1 * *», який планує виконання завдання 1 числа кожного місяця о 9 годині ранку за місцевим часом.

У додатку Е наведено реалізацію іншої функції Azure для оновлення статусу рахунків, яка є функцією зворотнього виклику або вебхуком. Вебхук (webhook) – це спосіб для веб-застосунків отримувати сповіщення в реальному часі про події, що відбуваються в інших застосунках. Тобто кожен раз, коли

стороння система платежів оновлюватиме статус рахунку, вона надсилатиме сповіщення на цю функцію і статус рахунку оновлюватиметься в нашій системі.

Для того, щоб інтегрувати OAuth2 в Swagger необхідно додати його до сервісів нашого додатку та виконати деякі налаштування.

```
builder.Services.AddSwaggerGen(options =>
{
    options.AddSecurityDefinition("oauth2", new()
    {
        Type = SecuritySchemeType.OAuth2,
        Flows = new OpenApiOAuthFlows
        {
            AuthorizationCode = new OpenApiOAuthFlow
            {
                AuthorizationUrl =
                  new Uri($"{oAuth2.Authority}/connect/authorize"),
                TokenUrl =
                  new Uri($"{oAuth2.Authority}/connect/token"),
                Scopes = oAuth2.Scopes
            }
        }
    });
    options.OperationFilter<AuthorizeCheckOperationFilter>();
});
```

Цей код налаштовує Swagger для відображення документації API з автентифікацією OAuth2. Він визначає схему безпеки OAuth2, та який потік OAuth2 використовувати. В цьому випадку використовується потік коду авторизації (AuthorizationCode). Для інтеграції з сервером авторизації, вказуються URL-адреси авторизації та отримання токена, а також області OAuth2. Крім того, в цьому коді додається фільтр операцій, який перевіряє авторизацію перед виконанням кожної операції API. Якщо поступить запит від анонімного користувача фільтр поверне 401 (Unauthorized) статус-код (http status). А на запити від користувачів, що не мають необхідних прав доступу – 403 (Forbidden), що забороняє доступ до ресурсу.

4 ОПИС ПРИЙНЯТНИХ ІНЖЕНЕРНИХ РІШЕНЬ

4.1 Реалізація функції створення розкладу за допомогою ШІ

Для створення розкладу дня та харчування для дітей було використано штучний інтелект (ШІ), а саме Gemini – модель генеративного ШІ від Google. Gemini була створена так, щоб бути мультимодальною, це означає, що вона може узагальнювати та бездоганно розуміти, оперувати та поєднувати різні типи інформації, включаючи текст, код, аудіо, зображення та відео. Окрім цього Gemini розуміє понад 40 мов, включаючи українську, що значно покращує користувальницький досвід та робить її зручною у використанні для користувачів, незалежно від мови, на якій вони спілкуються [12].

Для того, щоб почати використовувати Gemini API на Google Cloud Platform необхідно увімкнути API, яке надає Gemini доступне через Vertex AI API (див. рис. 4.1). Vertex AI – це платформа машинного навчання (ML) від Google, яка забезпечує єдине середовище для розробки, навчання та розгортання моделей ML. Вона пропонує різноманітні інструменти та функції, які спрощують процес роботи з ML, незалежно від рівня досвіду [13].

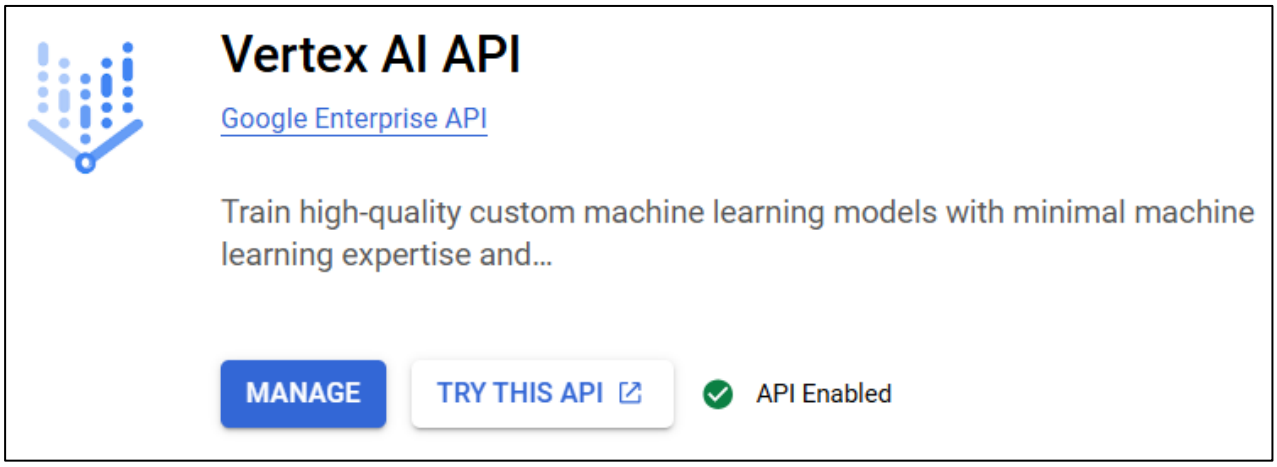


Рисунок 4.1 – Сторінка налаштування Vertex AI API (рисунок виконаний самостійно)

Після цього було створено сервісний акаунт (Service Account), що є особливим типом облікового запису в Google Cloud Platform (GCP), який не

пов’язаний з людиною-користувачем. Він використовується для надання дозволів програмам, скриптам або іншим автоматизованим системам, які не можуть авторизуватись з використанням людських облікових даних. В нашому випадку цей акаунт за допомогою згенерованого набору ключів доступу (JSON-файли), які використовуються для автентифікації, матиме доступ до Vertex AI API та зокрема Gemini API з розробленого API нашої програмної системи.

Нижче наведено код конструктору класу GoogleChatSession, повний код якого наведено в додатку Ж.

```
public GoogleChatSession(
    ILogger<GoogleChatSession> logger,
    IOptions<GoogleVertexAIConfiguration> vertexAiOptions
)
{
    var vertexAi = vertexAiOptions.Value;
    _logger = logger;
    _modelPath = $"projects/{vertexAi.ProjectId}/" +
                 $"locations/{vertexAi.Location}/" +
                 $"publishers/{vertexAi.Publisher}/" +
                 $"models/{vertexAi.Model}";
    _predictionServiceClient = new PredictionServiceClientBuilder
    {
        CredentialsPath = vertexAi.CredentialsPath,
        Endpoint = $"{vertexAi.Location}-aiplatform.googleapis.com",
        Logger = logger
    }.Build();
    _contents = new List<Content>();
}
```

Цей код ініціалізує клас для взаємодії з Vertex AI, для цього він вказує шлях до моделі та шлях до згенерованого ключа доступу, що є JSON-файлом та який використовуються для автентифікації в GCP.

4.2 Реалізація автоматичних рахунків для батьків

Кожного місяця для батьків, чиї діти знаходяться в дитячих садках, створюються рахунки за користування послугами садка на кожну дитину. В цих рахунках батьки повинні бачити наступну інформацію:

- ім’я дитини, на яку створено рахунок;
- інформацію про дитячий садок та групу;

– період (місяць), за який створено рахунок;
– детальна інформація про те, які послуги та в якій кількості включені в даний рахунок.

Для реалізації було використано дві Azure Functions: одну з тайм-тригером та одну з http-тригером (див. рис. 4.2).

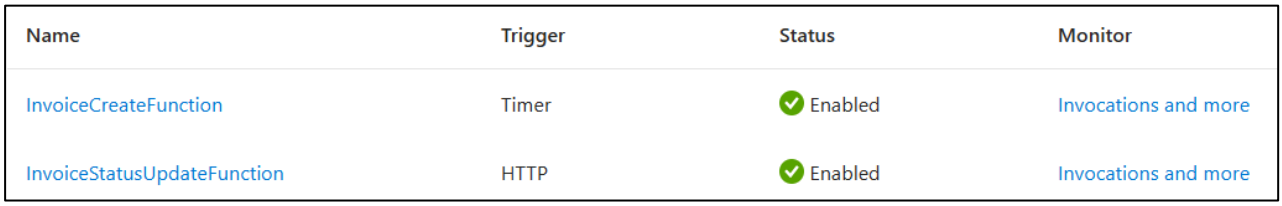


Рисунок 4.2 – Створені Azure Functions для рахунків (рисунок виконаний самостійно)

Ці дві функції виконують різні задачі:

– функція з тайм-тригером за відповідним розкладом, що має вигляд CRON-виразу [11], створює рахунки в сторонній платіжній системі та відповідно в нашій системі для збереження інформації про рахунки;
– функція з http-тригером реалізує функціонал вебхука (webhook), що отримує запити про зміну статусу рахунка від сторонньої платіжної системи та оновлює статус в нашій системі.

В якості сторонньої платіжної системи було обрано Monobank. Він надає своїм клієнтам зручний API, що може працювати в режимі тестового середовища. Для цього потрібно використовувати токен із https://api.monobank.ua/. Для тестування не потрібно мати термінал або дозвіл для тестування еквайрингу, тестове середовище доступне всім клієнтам банку. В тестовому середовищі для оплати можна використовувати будь-які номер, дату та cvv картки, навіть якщо вони не є дійсними. Єдина умова: номер картки повинен бути валідним за алгоритмом Луна. При використанні даних реальної банківської картки, вона буде прийнята, але фінансова авторизація здійснюватись не буде [14].

На рисунку 4.3 наведено сторінку сплати рахунку в системі Monobank.

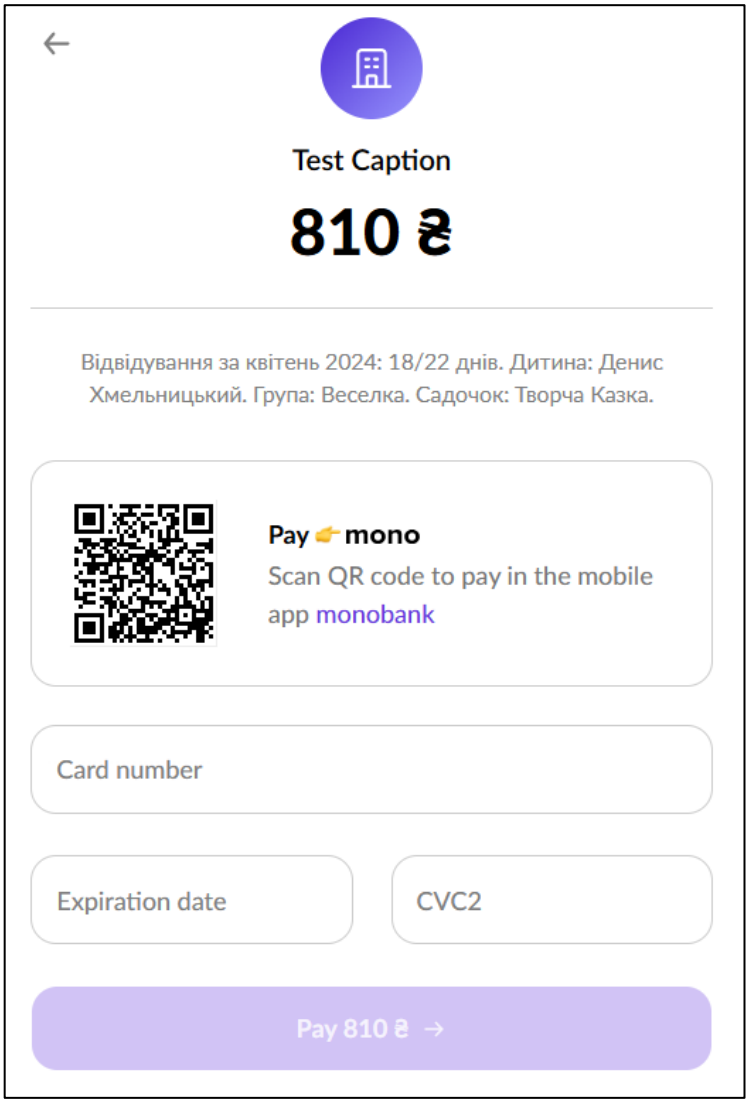


Рисунок 4.3 – Сторінка сплати рахунку в Monobank (рисунок виконаний самостійно)

На кожну зміну статусу рахунку серверна частина еквайрингу Monobank виконає до 3-х спроб POST запиту на адрес функції з http-тригером, що слугує вебхуком, поки у відповідь не отримає http-статус 200 OK. З параметром X-Sign у заголовках. Параметр повинен містити підпис тіла запиту вебхука по стандарту ECDSA [14]. Це потрібно для перевірки цілісності транзакції та запобігання підробкам. Код створення підпису наведено нижче.

```
private string CreateSignature(string message)
{
    using var sha256 = SHA256.Create();
    byte[] hash = sha256
```

```
            .ComputeHash(Encoding.UTF8.GetBytes(message));
        return Convert.ToBase64String(hash);
    }
```

Функція CreateSignature обчислює хеш-підпис для заданого рядка, який в нашому випадку є JSON-рядком тіла запиту вебхука, за допомогою алгоритму SHA256 і повертає його у вигляді рядка, закодованого в base64.

4.3 Реалізація повідомлень на пошту

Повідомлення на пошту були реалізовані за допомогою Azure Communication Service – сервісу, що надає зручний спосіб надсилання електронних листів за допомогою Azure. Для цього було створено доменне ім’я та під’єднано його до основного сервісу Azure, що відповідає за відправку листів (див. рис. 4.4). Це доменне ім’я буде використовуватися для відправки листів.

Рисунок 4.4 – Створене доменне ім’я на Azure (рисунок виконаний самостійно)

Нижче наведено метод, що відповідає за відправку повідомлень на електронну пошту, а в додатку И наведено повний код класу AzureCommunicationEmailSender.

```
_emailClient.Send(
    WaitUntil.Completed,
    senderAddress: _configuration.Sender,
    recipientAddress: email,
    subject: subject,
    htmlContent: htmlMessage);
```

Цей код отримує в параметрах адрес електронної пошти отримувача, тему листа та вміст листа, що може містити html-контент, й відправляє цього листа з адреси використовуючи змінну _configuration.Sender.

4.4 Реалізація додаткового захисту за допомогою QR-кодів

Для створення додаткового шару захисту було реалізовано тимчасові QR-коди, що генеруються батьками, а вихователі мають змогу їх сканувати для відмічання присутності дітей, або для відмітки, що їх забрали з садка. Це дозволить контролювати доступ, що забезпечить передачу дітей лише в руки авторизованих осіб, оскільки для кожної дитини створюється власний QR-код і інша особа, яка не має доступу до дитини не зможе згенерувати цей код і відповідно не зможе забрати її з садка.

В QR-коді закодовані дані у форматі JWT (Json Web Token), що містять інформацію про дитину, батька, який згенерував QR-код, та дію, для якої згенеровано код: привести дитину чи забрати її. Цей JWT токен використовується для створення масиву байтів, який формує зображення QR-коду. Такий підхід забезпечує надійність та безпеку передачі інформації, оскільки JWT токени зазвичай підписані цифровими підписами, що гарантує їхню автентичність та цілісність. Завдяки цьому QR-код може бути легко відсканований та розшифрований нашою системою, забезпечуючи швидкий та безпечний доступ до необхідних даних. Нижче наведено код для генерації зображення QR-коду.

```
private static byte[] GenerateQrCodeImage(string encryptedData)
{
    var qrGenerator = new QRCodeGenerator();
    var qrCodeData = qrGenerator
        .CreateQrCode(encryptedData, QRCodeGenerator.ECCLevel.Q);
    var qrCode = new PngByteQRCode(qrCodeData);
    return qrCode.GetGraphic(20);
}
```

Цей код створює екземпляр класу QRCodeGenerator, який відповідає за створення даних QR-коду. Він генерує дані з використанням вхідного рядка encryptedData та рівнем виправлення помилок, який визначає кількість зайвої інформації, що додається до QR-коду для підвищення стійкості до пошкоджень. В даному випадку використовується QRCodeGenerator.ECCLevel.Q, що

забезпечує високий рівень надійності. Метод GetGraphic генерує зображення QR-коду та повертає його у вигляді масиву байтів.

4.5 Розробка власного OAuth сервісу та його адміністрування

За допомогою популярної .NET бібліотеки IdentityServer4 було створено власний OAuth сервіс, що надає централізоване місце для управління обліковими даними користувачів. Це значить що веб, мобільним та десктопним додатками не потрібно розробляти власні сторінки входу, реєстрації та управління акаунтом користувачів.

На рисунку 4.5 наведено сторінку входу в систему, що надається сервером автентифікації та авторизації ChildWell Identity. На цій сторінці користувачам доступні можливості входу через вже існуючий локальний обліковий запис або за допомогою зовнішнього, що надає Google. При використанні входу за допомогою Google, користувача буде перенаправлено на сторінку входу до власного акаунту Google, після чого буде запитано дозвіл надати інформацію про ім’я, прізвище, по батькові та адресу електронної пошти користувача нашому додатку, після підтвердження якого його буде перенаправлено назад до нашого застосунку.

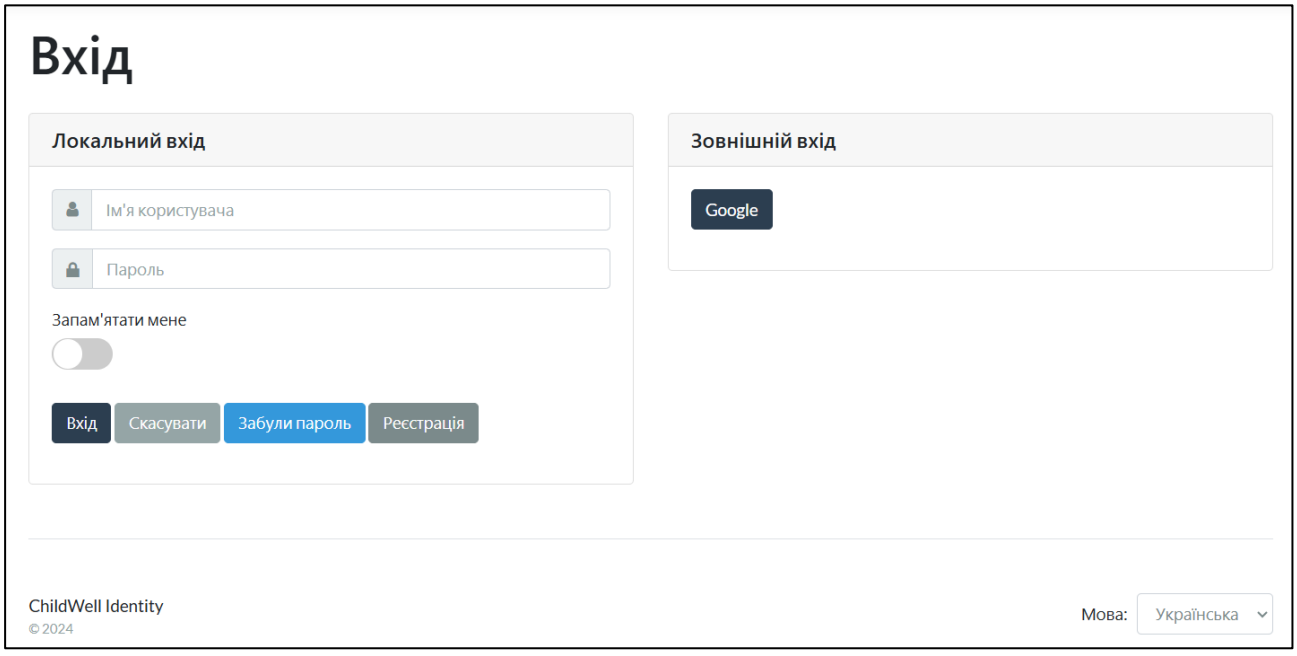


Рисунок 4.5 – Сторінка входу в систему (рисунок виконаний самостійно)

Розроблений сервер ChildWell Identity, надає користувачам важливий функціонал для управління власний акаунтом, а саме:

– управління правами доступу, які вони надали стороннім програмам;
– зміна паролю;
– перегляд та зміна даних власного профілю, що включає: ім’я, прізвище, по батькові, адресу електронної пошти, номер телефону та стать;
– видалення власного акаунту;
– можливість встановити або видалити двофакторну автентифікацію.

На рисунку 4.6 наведено головну сторінку ChildWell Identity.

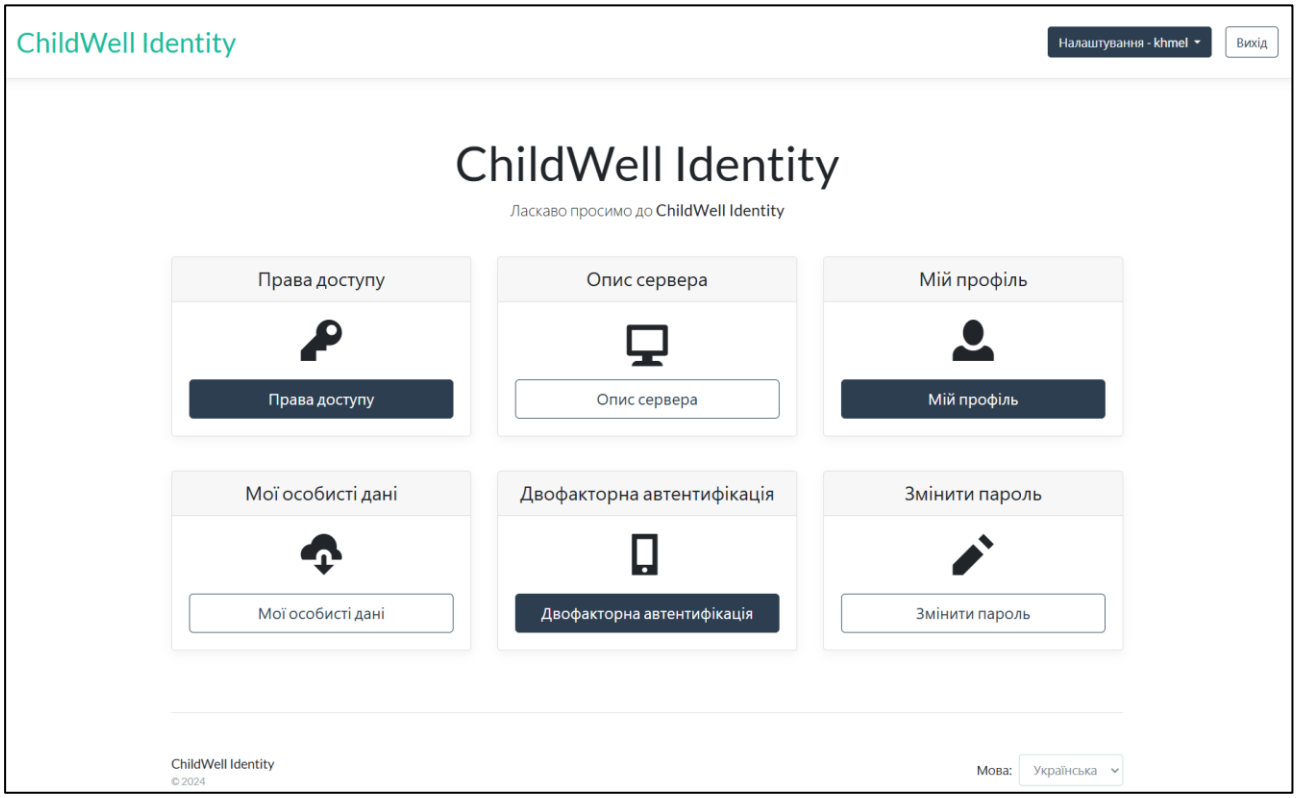


Рисунок 4.6 – Головна сторінка ChildWell Identity (рисунок виконаний самостійно)

Бібліотека IdentityServer4 також надає можливості адміністрування, що дозволяє додавати, видаляти та оновлювати користувачів, а також налаштовувати їхні облікові записи, присвоювати ролі та встановлювати паролі. Крім цього адміністрування дозволяє додавати, видаляти та оновлювати програмних клієнтів, надавати їм права доступу до певних API та ресурсів. Не

менш важливим є управління ролями та дозволами, що дозволяє створювати та керувати ролями, а також призначати користувачів до певних ролей. Це дозволяє легко надавати доступ до ресурсів на основі ролей, що спрощує управління доступом.

На рисунку 4.7 наведено головну сторінку ChildWell IdentityAdmin, що надає можливості адміністрування для ChildWell Identity.

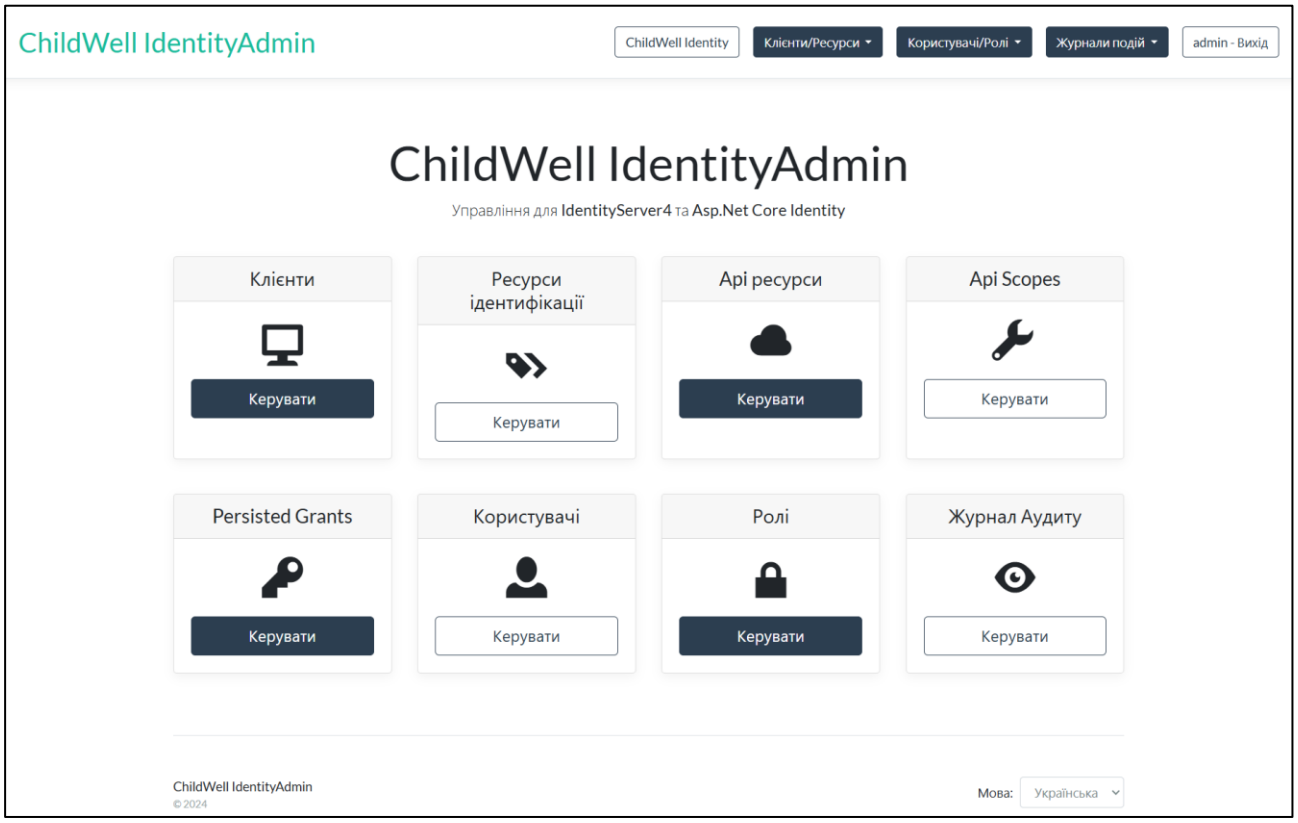


Рисунок 4.7 – Головна сторінка адміністрування для ChildWell Identity (рисунок виконаний самостійно)

Адміністрування для IdentityServer є важливим для забезпечення безпечного, ефективного та керованого доступу до наших ресурсів. Це дозволяє контролювати всі аспекти IdentityServer, від користувачів та програмних клієнтів до ресурсів і дозволів.

4.6 Реалізація реєстрації та автентифікації за допомогою Google

Ми вже маємо власний сервер автентифікації, тепер необхідно дозволити йому використовувати Google автентифікацію. Будь-яка програма (в нашому

випадку власний IdentityServer), що використовує OAuth2 для доступу до Google API, повинна мати облікові дані авторизації, які ідентифікують програму на сервері Google OAuth2. Для цього в консолі GCP необхідно перейти на сторінку облікових даних та створити новий ідентифікатор клієнта OAuth (OAuth client ID). На рисунку 4.8 наведено налаштування нового OAuth клієнта.

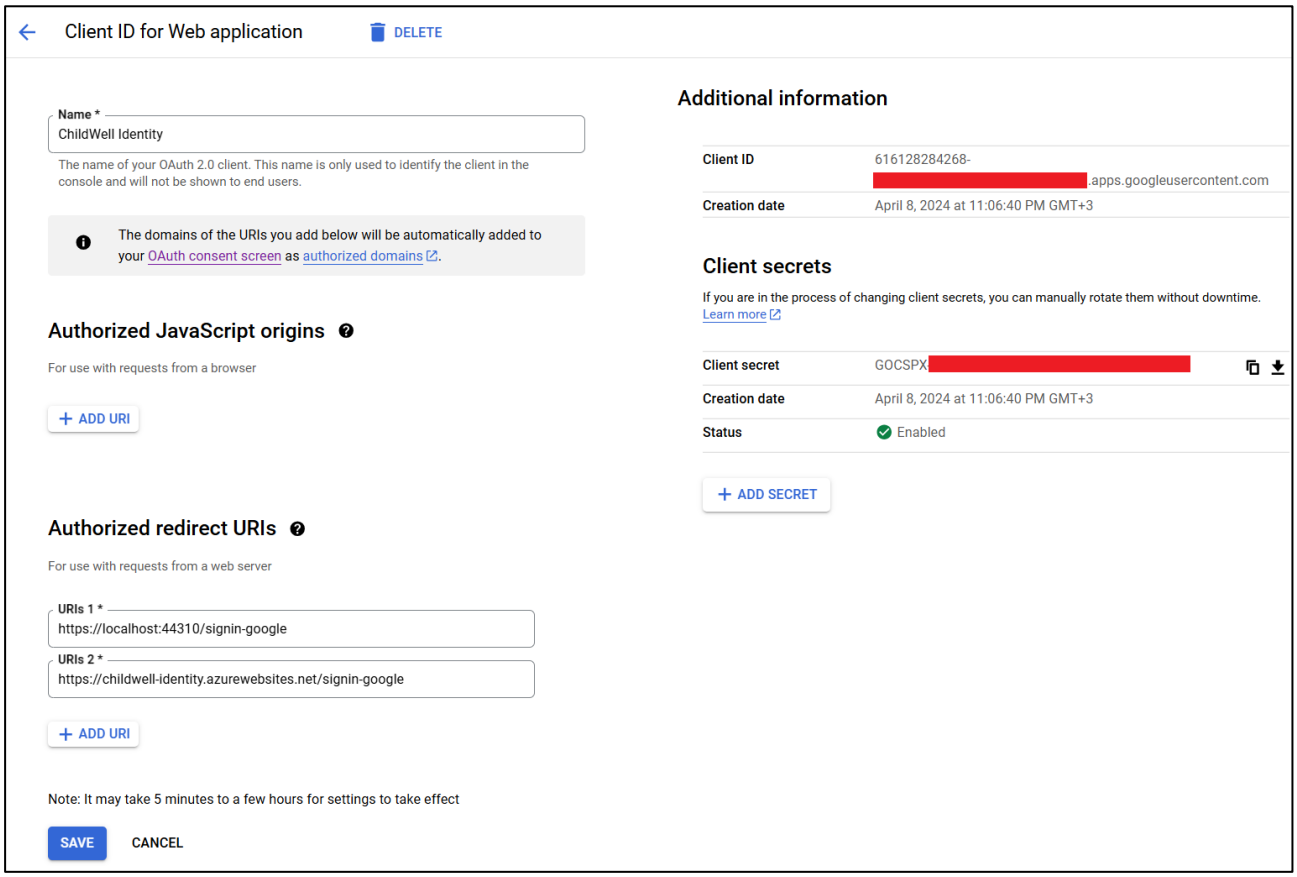


Рисунок 4.8 – Налаштування нового OAuth клієнта в консолі GCP (рисунок виконаний самостійно)

В налаштуваннях необхідно вказати тип додатку, який ми хочемо зареєструвати, в нашому випадку це веб-застосунок, та вказати його ім’я. Для веб-застосунків ще обов’язково потрібно вказувати авторизовані URI перенаправлення (Authorized redirect URIs). URI перенаправлення – це URL-адреса, на яку користувач перенаправляється після успішної автентифікації. Вони використовуються в контексті OAuth2 та інших протоколів автентифікації для забезпечення безпечного та надійного переходу користувача назад до програми або веб-сайту, що розпочала процес автентифікації.

Після успішного додавання клієнта, ми отримаємо його унікальний ідентифікатор та секрет, які треба використовувати при налаштуванні автентифікації Google в нашому додатку. Нижче наведено код такого налаштування.

```
authenticationBuilder.AddGoogle(options =>
{
    options.ClientId =
        externalProviderConfiguration.GoogleClientId;
    options.ClientSecret =
        externalProviderConfiguration.GoogleSecret;
    options.CallbackPath =
        externalProviderConfiguration.GoogleCallbackPath;
});
```

Цей код налаштовує автентифікацію Google для нашої системи, використовуючи інформацію про ідентифікатор клієнта та секрет, отримані при реєстрації його в консолі Google, та які зберігаються в конфігурації нашої системи та доступні зі змінної externalProviderConfiguration. Після того, як користувач успішно авторизується через Google, наш додаток отримає відповідь за допомогою шляхів зворотного виклику (callback), що зазначено у змінній GoogleCallbackPath.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Важливим етапом перед запуском системи є проведення ґрунтовного тестування, яке охопить усі аспекти її функціональності, забезпечуючи виявлення та виправлення можливих помилок, перевірку стабільності роботи та підтвердження відповідності системи всім вимогам і очікуванням користувачів.

Розроблена програмна система складається з серверної частини, яка тісно взаємодіє з:

- платіжною системою Monobank для прийому платежів від батьків;
- Azure Functions, відповідальними за автоматичне формування рахунків та оновлення їх статусу;
- Google Vertex AI для генерації розкладу дня та харчування для дітей;
- власним OAuth2 сервісом (IdentityServer) для автентифікації користувачів з різними ролями (адміністратори, вихователі, батьки).

Наявність різних модулів та їх взаємодія вимагає комплексного підходу до тестування. Окреме тестування кожного модуля забезпечить виявлення помилок в логіці його функціонування, але не гарантує стабільної роботи системи в цілому. Тому розглянемо стратегію, що включає тестування як окремих модулів, так і їх інтеграції.

5.1 Тестування окремих модулів системи

Модульне тестування зосереджувалося на ретельній перевірці окремих компонентів системи ізольовано від інших частин, з метою підтвердження їх відповідності специфікаціям та коректного виконання всіх функцій.

Тестування серверної частини акцентувалося на перевірці основних функціональностей API, забезпечуючи їхню коректну роботу без врахування інтеграції з іншими компонентами системи. Тестування валідації вхідних даних, було націлене на те, щоб перевірити чи API правильно валідує вхідні дані та повертає відповідні повідомлення про помилки у випадку некоректних даних. Обробка виключних ситуацій, була не менш важливим сценарієм перевірки, що

включало тестування обробки виключних ситуацій, наприклад, запитів з відсутніми або некоректними даними, а також відповіді сервера у випадку внутрішніх помилок. Що також включало перевірку на відповідність статусів HTTP, наприклад, чи повертає API правильні HTTP статуси для різних типів відповідей, наприклад, 200 для успішних запитів, 400 для помилок валідації, 401 для неавторизованих запитів і 500 для внутрішніх помилок сервера.

Окремо було протестовано розроблену функцію (Azure Function), що створює рахунки в нашій системі. Це включало перевірку правильності створення рахунків за розкладом та наявність необхідної інформації. Всі рахунки обов'язково повинні містити інформацію про дитину, садочок, групу, період та за яку послугу було виставлено рахунок. Ця інформація також повинна бути присутньою в рахунках, створених в Monobank. Крім цього, при наявності рахунків чий термін сплати вже сплив, цей рахунок повинен включатися в рахунок в наступному місяці і містити його інформацію разом з новою.

Для генерації розкладу було проведено тестування Google Vertex AI, що включало покращення запиту (промпту) до моделі штучного інтелекту, а також перевірку формату вихідних даних, які очікувались у форматі JSON. Крім цього було перевірено точність та якість згенерованого розкладу, дотримання критеріїв, що дозволять створювати продуктивні розклади для дітей.

Тестування OAuth2 сервісу (IdentityServer) перевіряло функції автентифікації (вхід в систему, зміна пароля, зміна персональних даних, відновлення доступу, правильна ідентифікація користувача), а також функція встановлення двофакторної автентифікації.

Результати тестування показали, що всі модулі серверної частини відповідають специфікаціям, обробка даних здійснюється коректно, а система стабільно працює при різних сценаріях використання, включаючи обробку виключних ситуацій. Крім цього для кожного модуля було проведено тестування інтернаціоналізації, яке підтвердило, що зміна мови не впливає на функціональність системи, забезпечуючи зручну та зрозумілу взаємодію для користувачів, які говорять різними мовами.

5.2 Інтеграційне тестування модулів системи

Інтеграційне тестування перевіряло взаємодію між різними модулями та сторонніми системами, щоб забезпечити їхню коректну сумісну роботу. Основна мета цього етапу – виявлення проблем, які можуть виникнути під час взаємодії компонентів, навіть якщо окремі модулі працюють правильно.

Сценарій інтеграції з платіжною системою Monobank перевіряв створення рахунків в Monobank та збереження їхніх ідентифікаторів в нашій системі, що ставали невід’ємною частиною внутрішніх рахунків. Було перевірено наявність необхідної інформації, що повинна дублювати усю інформацію, яка присутня у внутрішніх рахунках системи. Крім цього було перевірено зміну статусу рахунка при сплаті або закінчення його терміну дії. Цей статус повинен вчасно та коректно оновлюватися в нашій системі, відповідно до змін в Monobank, що і було підтверджено під час тестування.

Тестування взаємодії з Google Vertex AI перевіряло, як створений розклад зберігається в нашій системі після обробки та перетворення його у відповідний формат, що використовується нашою системою. Це забезпечило коректне інтегрування розкладу в загальний функціонал, підтвердивши, що дані успішно зберігаються і використовуються без помилок.

Тестування сценаріїв автентифікації для всіх ролей (адміністраторів, вихователів та батьків) підтвердило коректність наданих прав доступу та доступу до відповідних функцій і ресурсів для кожної ролі. Таким чином, система гарантує безпеку та належний рівень доступу для всіх користувачів відповідно до їхніх ролей.

Загалом, тестування системи пройшло успішно і підтвердило, що вона функціонує згідно з очікуваннями, забезпечуючи надійну роботу всіх її компонентів як окремо, так і в інтеграції один з одним. Тестування дозволило виявити та виправити деякі помилки, що сприяло підвищенню стабільності та оптимізації роботи системи. Результати тестування свідчать про високу якість і готовність системи до реального використання.

6 ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розроблена серверна частина є лише складовою великої комплексної системи, що складається з серверної частини, веб та мобільного застосунку.

6.1 Презентація стейкхолдерам

На зустрічі зі стейкхолдером наша комплексна програмна система для дитячих садків отримала значну увагу та високу оцінку. Стейкхолдер високо оцінив можливості системи щодо покращення освітніх процесів та підвищення ефективності роботи адміністративного та педагогічного персоналу. Зокрема було акцентовано увагу на інноваційних аспектах системи, таких як генерація розкладу за допомогою ШІ та автоматичне створення рахунків для батьків.

Стейкхолдери зазначили, що система має значний потенціал для впровадження у широкому масштабі, підкреслюючи її важливість для сучасної освітньої інфраструктури. Вони висловили впевненість у тому, що наша розробка стане важливим інструментом у щоденній діяльності дитячих садків, сприяючи підвищенню якості освітніх послуг та комунікації з батьками.

Позитивні відгуки та висока оцінка від стейкхолдерів на цій зустрічі підтверджують правильний напрямок нашої роботи та надихають на подальше вдосконалення системи. Це також відкриває нові можливості для співпраці та поширення нашого рішення, що підвищує його вплив та ефективність у сфері дошкільної освіти.

6.2 Публікація наукових тез

Наукова стаття під назвою «Аналіз використання штучного інтелекту для генерації розкладу занять і харчування дітей в закладах дошкільної освіти» [15], що була опублікована в рамках 28-го Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті», детально висвітлює аналіз використання штучного інтелекту в нашій системі, що дозволяє вчителям створювати розклади, адаптовані до потреб дошкільних груп і моделей

харчування, гарантуючи, що вони отримають правильний розклад і харчові звички. Батьки в свою чергу можуть легко бачити повсякденні дії та харчування своїх дітей, що полегшує отримання інформації про їх самопочуття.

Ця публікація відкриває нові можливості для академічного співробітництва та обміну досвідом, а також підкреслює важливість впровадження інноваційних рішень для покращення освітніх процесів.

6.3 Презентація на форумі

Ця комплексна програмна система була представлена в рамках 28-го Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті», де здобула високе визнання серед фахівців галузі та була нагороджена дипломом II ступеня, який наведено в додатку К.

На форумі, який зібрав провідних експертів у сфері інформаційних технологій та освіти, наша комплексна програмна система для дитячих садків привернула значну увагу завдяки своїй інноваційності та практичній корисності. Система була високо оцінена за її здатність забезпечувати безпечну автентифікацію користувачів та автоматизувати ключові процеси управління.

Особливо відзначено інтуїтивно зрозумілий інтерфейс системи, який спрощує адаптацію користувачів, а також її гнучкість та масштабованість, що дозволяє налаштовувати систему відповідно до потреб різних дитячих садків. Журі форуму визнало систему однією з найкращих у своїй категорії, присудивши їй диплом II ступеня.

Ця нагорода є важливим підтвердженням якості та ефективності нашої розробки. Вона підкреслює значення інноваційних рішень для поліпшення освітніх процесів і надихає нас на подальше вдосконалення системи. Визнання, отримане в рамках 28-го Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті», відкриває нові можливості для співпраці та розширення використання нашої системи в дитячих садках по всій країні та за її межами.

ВИСНОВКИ

Згідно з результатами аналізу, у даній роботі була розроблена серверна частина комплексної програмної системи для дитячих садків, яка цифровізує процеси закладу та забезпечує необхідним та зручним функціоналом батьків, вихователів та адміністраторів закладів, що вирішує їхні проблеми з якими вони стикаються кожен день. Розробка базувалась на використанні сучасних технологій, включаючи штучний інтелект (ШІ) та хмарні обчислення. Крім того було запроваджено розширену систему безпеки, використовуючи тимчасові QR-коди, протокол авторизації OAuth2 та двофакторну автентифікацію.

Розроблена програмна система вирізняється наявністю унікальних функцій, що відповідають на потреби цільової аудиторії, що може вже казати про її успішність. Система впроваджує функції для автоматизації створення розкладу дітей за допомогою ШІ, що значно економить час та зусилля персоналу. Крім цього система надає зручну реєстрацію дітей та подання заяв до садків, відстеження присутності дітей та використання QR-кодів для , планування режиму дня з можливістю створення власних занять, рахунки для батьків по сплаті за послуги дитячих садків – все це робить програмну систему конкурентоспроможним гравцем на ринку.

Результати даної роботи мають велике значення та кладуть початок в інтеграції сучасних технологій в заклади дошкільної освіти з метою підвищення ефективності їх роботи та забезпечення оперативного та прозорого зв’язку між батьками та закладами.

Функціонал розробленої системи може бути розширений шляхом інтеграції з іншими рішеннями, наприклад документообігу, управління підприємствами, а також навчальних програм з елементами геймфікації для інтерактивного навчання дітей, що об’єднає ще більше можливостей в одній системі та зможе вирішувати ще більше проблем, які постають перед закладами дошкільної освіти.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. KidKare [Електронний ресурс] – URL: https://www.kidkare.com (дата звернення: 23.04.2024).

2. Brightwheel [Електронний ресурс] – URL: https://mybrightwheel.com (дата звернення: 23.04.2024).

3. EZChildTrack [Електронний ресурс] – URL: https://www.ezchildtrack.com (дата звернення: 23.04.2024).

4. Procare Software (Kinderlime) [Електронний ресурс] – URL: https://www.procaresoftware.com (дата звернення: 24.04.2024).

5. Lillio (HiMama) [Електронний ресурс] – URL: https://www.lillio.com (дата звернення: 24.04.2024).

6. Carlier, B. “An Introduction to OAuth2.0”, IDPro Body of Knowledge 1(12). doi: https://doi.org/10.55621/idpro.99, 2023.

7. R. Fielding. Architectural Styles and the Design of Network-based Software Architectures. Ph.d. dissertation, University of California, Irvine, 2007.

8. C. Pautasso. Restful web services: principles, patterns, emerging technologies. InWeb Services Foundations, pages 31–51. Springer, 2014.

9. Marcotte C.-H., Zebdi A. An atypical ASP.NET core 5 design patterns guide: A SOLID adventure into architectural principles, design patterns, .NET 5, and C#. Packt Publishing, 2020. 762 с.

10. Khan O. M. A., Senthilvel G., Qureshi H. A. Enterprise Application Architecture with .NET Core: An architectural journey into the Microsoft .NET open source platform. Packt Publishing, 2017. 564 с.

11. Mishra A., Satapathi A. Hands-On azure functions with C#: build function as a service solutions. Apress L. P., 2021. 514 p.

12. Introducing Gemini: our largest and most capable AI model [Електронний ресурс] – URL: https://blog.google/technology/ai/google-gemini-ai (дата звернення: 25.04.2024).

13. Bhatia J., Chaudhary K. The Definitive Guide to Google Vertex AI: Accelerate your machine learning journey with Google Cloud Vertex AI and MLOps best practices. Packt Publishing, 2023. 422 p.

14. Monobank API – Acquiring [Електронний ресурс] – URL: https://api.monobank.ua/docs/acquiring.html (дата звернення: 25.04.2024).

15. Радіоелектроніка та молодь у XXI столітті. Т. 6 : Конференція "Інформаційні інтелектуальні системи" : матеріали 28-го Міжнар. молодіж. форуму, 16–18 квіт. 2024 р. / М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. – Харків : ХНУРЕ, 2024. – 958 с. – DOI: 10.30837/IYF.IIS.2024.

ДОДАТОК А

Звіт з результатами перевірки на унікальність тексту в базі ХНУРЕ

UNICHECK
by Turnitin

Ім'я користувача:
Олійник Олена Володимирівна каф. ПІ

ID перевірки:
1016313876

Дата перевірки:
03.06.2024 10:38:40 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
03.06.2024 10:39:01 EEST

ID користувача:
100012353

Назва документа: 2024_Б_ПІ_ПЗПІ_20_1_Смєйко_Б_М_скорочений

Кількість сторінок: 52 Кількість слів: 8848 Кількість символів: 73404 Розмір файлу: 1.38 MB ID файлу: 1016111045

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

4.77%
Схожість

Найбільша схожість: 1.86% з джерелом з Бібліотеки (ID файлу: 1016107711)

Пошук збігів з Інтернетом не проводився

4.77% Джерела з Бібліотеки 230 ..... Сторінка 54

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування 15 сторінок

ДОДАТОК Б

Специфікація програмної системи

ChildWell

Software Requirements Specification

4.0

15.02.2024

Богдан Смєйко
Віталій Нестеренко
Анна Бондаренко

Історія версій

Дата	Опис	Автор	Коментарі
25.01.2024	Версія 1.0		Перша редакція
05.02.2024	Версія 2.0	Анна Бондаренко	Перелік функцій продукту розміщено в порядку пріоритетності Проєкту надано назву
10.02.2024	Версія 3.0	Богдан Смєйко	Оновлено класи
15.02.2024	Версія 4.0	Віталій Нестеренко	Оновлено діаграми

Затвердження документів

Наступну Специфікацію вимог до програмного забезпечення було прийнято та схвалено:

Підпис	Ім’я	Роль	Дата
	Богдан Смєйко	Back-end частина	15.02.2024
	Віталій Нестеренко	Mobile частина	15.02.2024
	Анна Бондаренко	Front-end частина	15.02.2024

Зміст

1. Вступ

У сучасному світі розвиток дитини визначається не лише академічними досягненнями, проте й фізичним та ментальним здоров’ям, емоційним станом, обсягом уваги. Тож виникає виклик у створенні та впровадженні ефективного розпорядку дня в закладах дошкільної освіти, де закладається фундамент для розвитку подальших здібностей дитини та формування їхніх звичок, що безумовно має вплив на все подальше життя вихованців. Важко сумістити бажання вихователів забезпечити оптимальний розвиток дітей, уникнути виснаження та гарантування засвоєння необхідних навичок і знань та потребу батьків у своєчасному доступі до інформацію про активності дітей в дитячих садках.

Виходячи з цього, постає актуальна потреба в розробці системи, яка не лише допоможе в розвитку навичок, а дозволить планувати розпорядок дня так, щоб діти не виснажувалися й забезпечить батькам надійний та миттєвий доступ до інформації щодо режиму дня та харчування їхніх дітей. Наша програма має на меті оптимізацію описаних вище процесів, що відповідає як вимогам педагогів, так і батьків. Детальніше про проєкт розповідається в даному документі.

1.1 Мета

Мета документу — надати детальний опис розроблюваної програмної системи для закладів дошкільної освіти з керуванням режимом дня та харчуванням дітей. Специфікація призначена для опису функціональних та нефункціональних вимог першої версії програмного продукту. SRS використовуватиметься членами команди, які будуть реалізовувати та забезпечувати коректну роботу системи та іншими стейкхолдерами.

1.2 Межі

Система, яку ми розробляємо отримала назву «ChildWell». Вона надаватиме можливість для викладачів створювати ефективний розклад для

вихованців з урахуванням їх фізіологічних особливостей, який гарантує здобуття потрібних навичок. Для батьків — можливість отримувати інформацію про активності дитини в закладі дошкільної освіти та її харчування в режимі реального часу.

Корисним функціоналом для вихователів стане планувальних розпорядку дня дітей, щоб вони не виснажувались та здобували потрібні навички. Батьки ж в свою чергу зможуть швидко отримувати достовірну інформацію щодо режиму дня дітей та їхнього харчування. Більш того корисним функціоналом стане можливість відмічання присутності дітей за допомогою QR-кодів, що буде доступним як для батьків так і вихователів. Крім цього батьки зможуть управляти заявками на подання дітей до закладів дошкільної освіти, а після матимуть змогу зручно сплачувати рахунки за послуги закладу.

Таким чином, система сприятиме ефективній організації розвитку дітей, забезпечуючи комфорт та контроль як для вихователів й адміністраторів закладу, так і для батьків.

1.3 Означення та абревіатури

Front-end — частина програмного забезпечення, яка відповідає за взаємодію користувача з системою або веб-додатком (дизайн, інтерфейс користувача та усі елементи, які користувач може бачити та з якими він може взаємодіяти).

int — дані цілочисельного типу.

text — дані текстового типу.

bool — це тип даних, який має одне з двох можливих значень (істине і хибне), які призначені для представлення двох істинних значень логіки та булевої алгебри.

date — тип даних для зберігання інформації про конкретний день у форматі рік-місяць-день.

datetime — тип даних, що включає в себе інформацію про конкретний момент у часі, включаючи як дату, так і час.

Git — це розподілена система контролю версій, яка відстежує зміни в будь-якому наборі комп'ютерних файлів.

ISO/IEC 27001 — це міжнародний стандарт для управління інформаційною безпекою.

WCAG (Інструкції щодо доступності веб-вмісту) — частина серії рекомендацій щодо доступності веб-сайтів, опублікованих Ініціативою веб-доступності (WAI) Консорціуму Всесвітньої павутини (W3C), головною міжнародною організацією стандартів для Інтернету. Вони являють собою набір рекомендацій щодо того, як зробити веб-вміст більш доступним.

GDPR (Загальний регламент захисту даних) — це нормативний акт Європейського Союзу щодо конфіденційності інформації в Європейському Союзі та Європейській економічній зоні.

1.4 Посилання

- ГОСТ 34.602-89: стандарти для автоматизованих систем, які регулюють вимоги до проектування та розробки.
- Методичні рекомендації МОН України: вказівки Міністерства освіти і науки України щодо організації дошкільної освіти дітей.
- ISO/IEC 27001: стандарт, що встановлює вимоги до систем управління інформаційною безпекою, який допоможе забезпечити безпеку даних користувачів.
- WCAG (Web Content Accessibility Guidelines): рекомендації щодо доступності веб-контенту, що допоможуть зробити додаток доступним для всіх користувачів, включаючи людей з обмеженими можливостями.
- GDPR (General Data Protection Regulation): регламент ЄС щодо захисту персональних даних.

1.5 Огляд

Цей документ містить повну інформацію про проєкт: його перспективи, функції, інтерфейс, обмеження, відомості про варіанти використання та діаграми для візуалізації різних аспектів системи. Кожна секція призначена для різних стейкхолдерів, від чого залежить зміст секцій.

Розділ «Загальний опис» націлений на опис загальних факторів, що впливають на вимоги до продукту і продукт в цілому. Цільовою аудиторією цього розділу перш за все є власник проєкту.

Розділ «Конкретні вимоги», на відміну від попереднього, містить подробиці щодо вимог до проєкту, його інтерфейси та функціональність, у ньому деталі реалізації описані технічною мовою для сприйняття розробниками.

У розділі «Моделі аналізу» розміщено діаграми та моделі, що використовуватимуться в процесі розробки відповідно до вимог, визначених в цьому документі SRS. Вони допоможуть у розумінні системи, шляхом візуалізації її роботи та взаємодії, що відбувається всередині системи.

Останній розділ, «Процес управління змінами», передбачує та описує яким чином відбуватиметься оновлення SRS, якщо виникне така необхідність при внесенні змін до вимог чи обсягів проєкту. Також він має вказувати на те, ким вноситимуться та ухвалюватимуться зміни.

2. Загальний опис

2.1 Перспективи продукту

Система планується як самостійний продукт, що не залежить від схожих застосунків. Для повноцінного використання користувачі повинні зареєструватися та використовувати браузерну або мобільну версію додатку.

2.2 Функції продукту

Основною метою продукту є оптимізація планування розпорядку дня дітей у закладах дошкільної освіти з управлінням харчуванням. Викладачам треба мати змогу керувати режимом дня дітей та отримувати інформацію про те, чи є він відповідним до вікових особливостей вихованців. Батьки ж у свою чергу отримають інструмент для відстеження активностей дитини та комунікації з працівниками закладу дошкільної освіти. До основних функцій продукту належать наступні:

- MF-1: Реєстрація та авторизація користувачів;
- MF-2: Створення оптимального розкладу для дітей, з урахуванням фізіологічних особливостей;
- MF-3: Зміна вихователями розкладу;
- MF-4: Контроль батьків над процедурою передачі дітей в довірені руки;
- MF-5: Автоматичне сповіщення вихователів про відсутність дитини;
- MF-6: Електронний журнал відвідуваності з часовими рамками;
- MF-7: Перегляд докладної інформації про режим дня та активності дитини;
- MF-8: Оплата рахунків (батьками);
- MF-9: Подача заявок до закладу освіти;
- MF-10: Керування заявками до закладу освіти;
- MF-11: Налаштування завідувачем (адміністратором) певних аспектів закладу освіти.

Також система повинна надавати інформацію про розробників та контакти.

2.3 Характеристики користувачів

Передбачено 3 види користувачів, що матимуть доступ до системи:

- батьки дітей дошкільного віку (подання та управління заявками до дитячих садків, перегляд режиму дня, сплата рахунків);
- вихователі закладів дошкільної освіти (створення режиму дня, відмічання присутності дітей);

- адміністративний персонал закладу (управління заявками від батьків, управління та налаштування дитячого садка, створення занять, реєстрація вихователів).

2.4 Загальні обмеження

Система повинна працювати на пристроях з наступними характеристиками:

- мінімум 4 ГБ оперативної пам'яті;
- процесор не нижче як чотирьохядерний, з тактовою частотою не менше 2.0 ГГц.
  Система має забезпечувати сумісність з операційними системами:
- Android 6.0 або новіше;
- iOS 13.4 або новіше.

Сучасні браузери підтримують виконання JavaScript, тому front-end частина буде виконана з використанням фреймворку React.js. Система повинна бути готовою до впровадження не пізніше 8 травня 2024 року. Система повинна відповідати стандартам безпеки даних, включаючи шифрування особистої інформації та заходи забезпечення конфіденційності.

2.5 Припущення й залежності

- Припущення: Користувачі мають доступ до Інтернету і він достатньо якісний;
    - Залежність: За відсутності доступу до Інтернету, користувачі не можуть використовувати застосунок у повній мірі;
- Припущення: Користувачі зареєстровані в системі;
    - Залежність: Незареєстровані користувачі не можуть користуватися системою в повному обсязі;

- Припущення: Всі користувачі мають апаратні та програмні засоби, необхідні для коректної роботи системи, зокрема, підтримують необхідні версії браузерів та операційних систем;
    - Залежність: Використання браузерів або пристроїв, що не підтримуються системою, може призвести до нестабільної роботи або відмови в доступі до певних функцій;
- Припущення: Всі користувачі володіють основами мови, в якій реалізовано інтерфейси користувача;
    - Залежність: Коректне користування системою передбачає знання користувачами мові, в якій вона реалізована.

3. Конкретні вимоги

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

Зареєстровані користувачі поділятимуться на вихователів та батьків й матимуть трохи різні інтерфейси.

Ці обидва типи користувачів матимуть доступ до режимів дня та планів харчування дітей, але лише вихователі зможуть їх редагувати. Також батьки матимуть окрему сторінку з дошкою оголошень, на якій бачитимуть повідомлення від дитсадків та вихователів.

3.1.2 Апаратний інтерфейс

Для відображення інтерфейсу користувача до пристрою має бути підключений дисплей.

3.1.3 Програмний інтерфейс

Для доступу до веб-системи користувач повинен використовувати веб-браузер, який повинен підтримувати Javascript, HTML5, CSS3. Для використання

мобільного додатку, користувачеві необхідно мати пристрій з ОС Android 6.0+ або iOS 13.4+.

3.1.4 Комунікаційні інтерфейси

Щоб користувач мав доступ до веб-системи, необхідна наявність Інтернету.

3.2 Функціональні вимоги

3.2.1 Реєстрація та авторизація користувачів

3.2.1.1 Опис

Нові користувачі можуть створювати акаунти в системі реєструватися за допомогою Google акаунта та використовувати їх для входу.

3.2.1.2 Вхідні умови

Заповнення усіх обов’язкових полів для реєстрації або входу в систему.

3.2.1.3 Виконання

Сервер створює нового користувача при реєстрації. При вході в систему сервер перевіряє чи існує такий користувач, якщо ні, то виводить відповідне повідомлення.

3.2.1.4 Вихідні умови

Створений новий користувач.

3.2.1.5 Обробка помилок

Якщо введено неправильний пароль або логін, буде показане відповідне повідомлення.

3.2.2 Генерація розкладу для дітей, з урахуванням фізіологічних особливостей

3.2.2.1 Опис

Система в автоматичному режимі може створити розклад для дітей, враховуючи розподіл часу між іграми, навчанням, прогулянками та прийомами їжі.

3.2.2.2 Вхідні умови

Задано початкові значення та пріоритети (обрано активності з наданих категорій).

3.2.2.3 Виконання

Сервер виконує обробку, розташовуючи активності в потрібному та правильному порядку.

3.2.2.4 Вихідні умови

Створений розклад дня для дітей.

3.2.2.5 Обробка помилок

В разі помилки система запропонує змінити пріоритети в розкладі, так як з наявними неможливо створити оптимальний розклад.

3.2.3 Зміна викладачами режимів дня та активностей

3.2.3.1 Опис

Викладачі можуть корегувати розклад дня дітей, що був згенерований системою.

3.2.3.2 Вхідні умови

Користувач повинен мати роль викладача.

3.2.3.3 Виконання

Сервер оновлює існуючий розклад.

3.2.3.4 Вихідні умови

Змінений розклад дня.

3.2.3.5 Обробка помилок

Якщо користувач не викладач – заборонити зміну розкладу.

3.2.4 Перегляд журналу відвідування групи

3.2.4.1 Опис

Вихователі повинні мати можливість переглядати журнал відвідуваності групи дітьми за обраний день.

3.2.4.2 Вхідні умови

Обрано групу та день, за який присутня інформація про відвідуваність групи дітьми.

3.2.4.3 Виконання

Сервер обробляє запит на надає відповідні дані у відповідь.

3.2.4.4 Вихідні умови

Журнал відвідуваності групи за обраний день успішно відображено на екрані вихователя.

3.2.4.5 Обробка помилок

В разі помилки система виводить повідомлення про відсутність запитуваних даних.

3.2.5 Контроль батьків над процедурою передачі дітей в довірені руки

3.2.5.1 Опис

Можливість батьків контролювати процедури передачі своєї дитини в довірені руки.

3.2.5.2 Вхідні умови

Батьки обирають дитину, для якої треба згенерувати check-in/check-out QR-код.

3.2.5.3 Виконання

Сервер генерує QR-код, що містить всю необхідну інформацію для передачі дитини в довірені руки.

3.2.5.4 Вихідні умови

Надано дані, які система буде використовувати при здійсненні цієї процедури.

3.2.5.5 Обробка помилок

Система надає повідомлення про будь-які помилки.

3.2.6 Автоматичне сповіщення вихователів про відсутність дитини

3.2.6.1 Опис

Система автоматично повідомляє вихователів про відсутність дитини, відображаючи відповідну інформацію в журналі присутності.

3.2.6.2 Вхідні умови

Система фіксує інформацію про відсутність дитини, якщо її присутність не зареєстровано за допомогою QR-код.

3.2.6.3 Виконання

Сервер автоматично збирає необхідну інформацію та надає її вихователям.

3.2.6.4 Вихідні умови

Вихователі отримують інформацію про відсутність дитини в закладі дошкільної освіти.

3.2.6.5 Обробка помилок

Система виявляє можливі конфлікти чи невідповідності в інформації і повідомляє вихователів для вирішення ситуації.

3.2.7 Додавання дитини в групу

3.2.7.1 Опис

Керівники ЗДО матимуть можливість додати дитину в групу, обравши одну з-поміж переліку груп садочка, відповідних до віку дитини.

3.2.7.2 Вхідні умови

Для додавання дитини в групу користувач повинен мати роль керівника ЗДО. Дитина як і група повинні бути обрані для коректного здійснення операції.

3.2.7.3 Виконання

Сервер додає нову дитину до групи та зберігає зміни.

3.2.7.4 Вихідні умови

Батьки та керівник бачать оновлену інформацію .

3.2.7.5 Обробка помилок

При відсутності необхідної ролі – повідомити користувача про це.

3.2.8 Перегляд інформації про режим дня та активності дитини

3.2.8.1 Опис

Батьки та вихователі зможуть переглядати детальну інформацію про режим дня та активності дітей.

3.2.8.2 Вхідні умови

На відповідній сторінці календар дозволить переглядати розклад і одному з режимів (місяць, тиждень, день, табличний вигляд).

3.2.8.3 Виконання

Сервер збирає та надсилає інформацію про дитину.

3.2.8.4 Вихідні умови

Батьки або вихователі бачать інформацію про режим дня та активності дитини.

3.2.8.5 Обробка помилок

При помилці сервер спробує зібрати інформацію кілька разів, після чого відобразить повідомлення про неуспішність операції.

3.2.9 Оплата рахунків для батьків

3.2.9.1 Опис

Функція дозволяє батькам здійснювати оплату рахунків за утримання та надані послуги в дошкільному закладі.

3.2.9.2 Вхідні умови

Батькам доступно отримання деталізації рахунків.

3.2.9.3 Виконання

Система формує рахунки для батьків, відображаючи вартість та послуги, надані дитині.

3.2.9.4 Вихідні умови

Батьки мають можливість вчасно та зручно оплачувати рахунки.

3.2.9.5 Обробка помилок

Система надає повідомлення про будь-які помилки та рекомендації для виправлення ситуації.

3.2.10 Подача заявки до закладу освіти

3.2.10.1 Опис

Батьки матимуть можливість подати заявку на вступ до закладу дошкільної освіти.

3.2.10.2 Вхідні умови

Обрано дитину, на ім’я якої й буде створено заявку, набір у освітній заклад відкрито.

3.2.10.3 Виконання

Сервер обробляє подану заявку, зберігає її та надає інформацію про неї керівнику ЗДО.

3.2.10.4 Вихідні умови

Можливість переглянути заявки на відповідній сторінці.

3.2.10.5 Обробка помилок

При помилці сервер спробує відправити заявку кілька разів, після чого відобразить повідомлення про неуспішність відправлення.

3.2.11 Керування заявкою до закладу освіти

3.2.11.1 Опис

Батьки матимуть можливість скасувати, прийняти або відхилити заявку. Керівники ЗДО матимуть можливість для перегляду деталей заявки, прийому або відхилення чи коментування заявки.

3.2.11.2 Вхідні умови

Обрано заявку, керування якою треба здійснити.

3.2.11.3 Виконання

Сервер обробляє оновлену заявку, зберігає її.

3.2.11.4 Вихідні умови

Можливість переглянути заявки на відповідній сторінці та оновити за допомогою відповідних кнопок.

3.2.11.5 Обробка помилок

При помилці сервер спробує оновити заявку кілька разів, після чого відобразить повідомлення про неуспішність відправлення.

3.2.12 Оновлення налаштувань активностей закладу освіти

3.2.12.1 Опис

Керівники ЗДО можливість оновити налаштування активностей ЗДО.

3.2.12.2 Вхідні умови

Обрано ЗДО та активності.

3.2.12.3 Виконання

Сервер обробляє оновлені налаштування, зберігає їх.

3.2.12.4 Вихідні умови

Можливість переглянути налаштування на відповідній сторінці та оновити за допомогою відповідних кнопок.

3.2.12.5 Обробка помилок

При помилці сервер спробує оновити налаштування кілька разів, після чого відобразить повідомлення про неуспішність відправлення.

3.3 Варіанти використання (Use Cases)

3.3.1 Неавтентифікований варіант використання

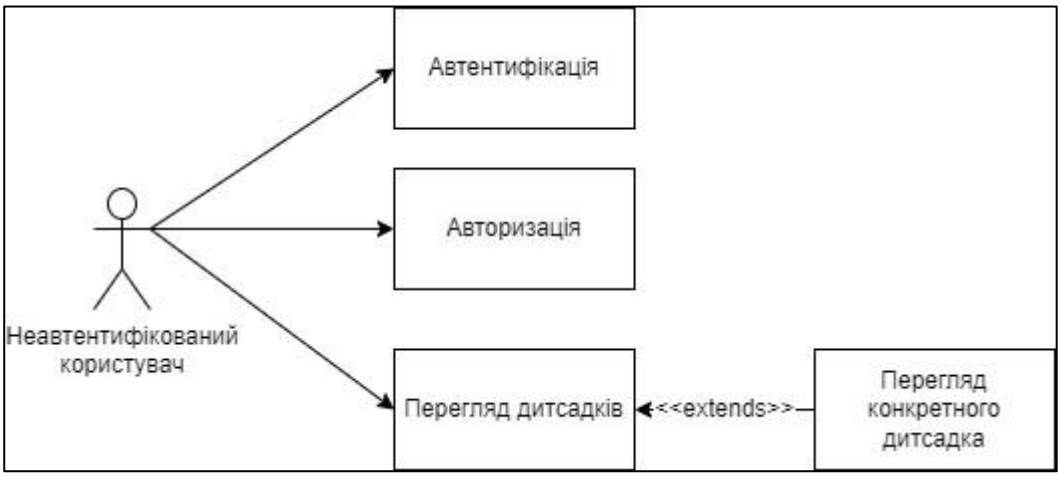


Рисунок 3.1 – Неавтентифікований варіант використання

3.3.2 Варіант використання батьків

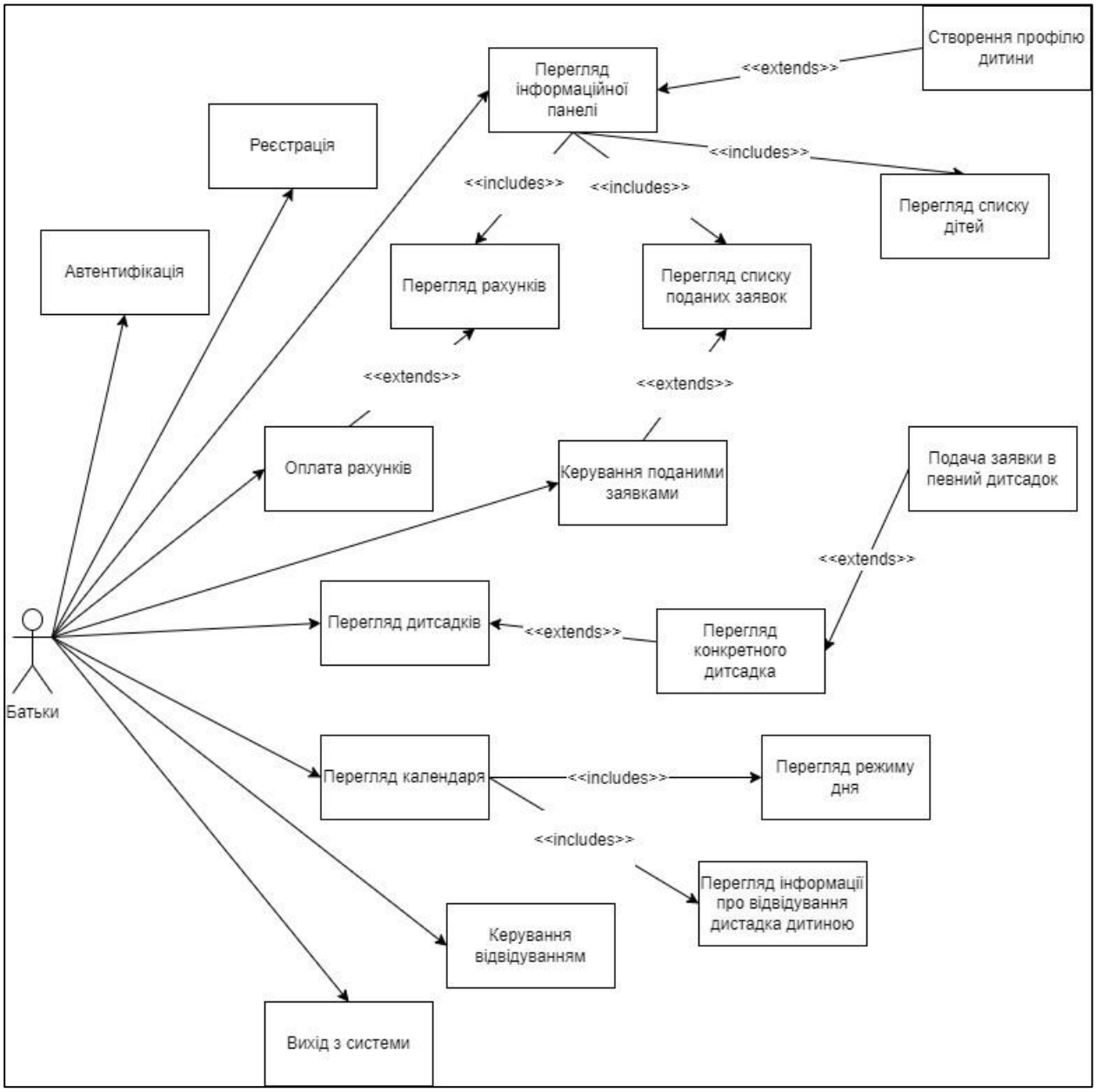


Рисунок 3.2 – Варіант використання батьків

3.3.2 Варіант використання вихователів

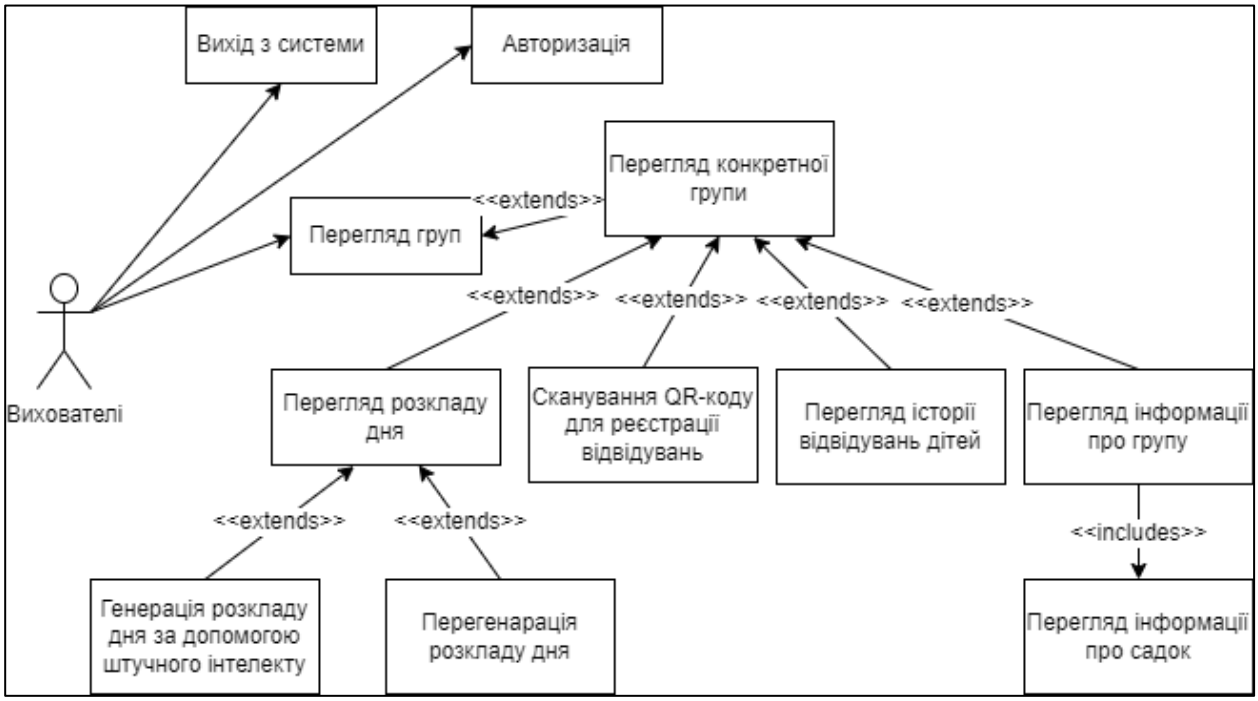


Рисунок 3.3 – Варіант використання вихователів

3.3.3 Варіант використання адміністраторів садочків

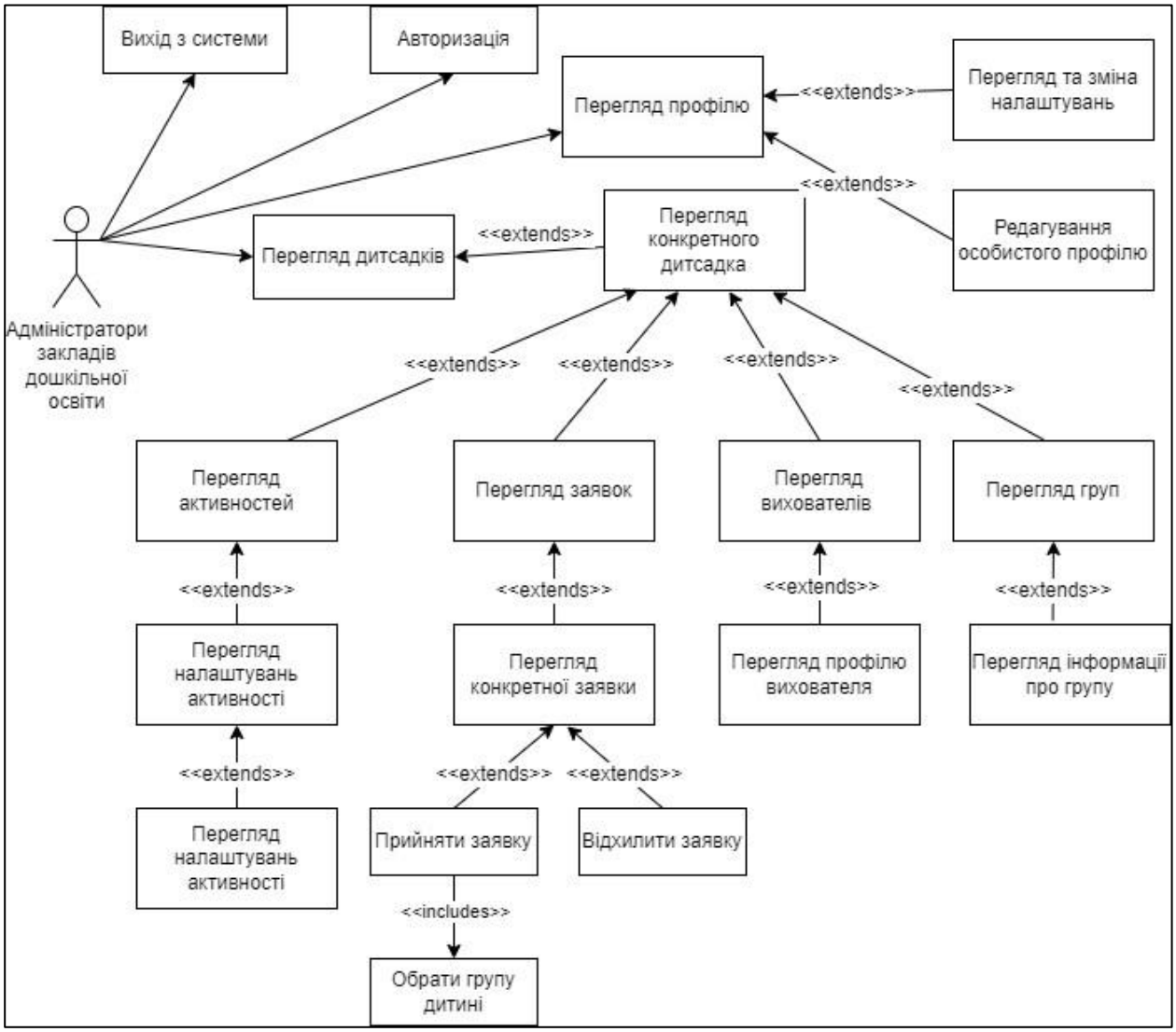


Рисунок 3.4 – Варіант використання адміністраторів садочків

3.4 Класи / Об’єкти

3.4.1 Group

3.4.1.1 Атрибути

Id – text

PreschoolId – text

Name – text

Description – text

MinAge – int

MaxAge – int

StartDateTime – datetime

EndDateTime – datetime

3.4.2 Child

3.4.2.1 Атрибути

Id – text

ParentId – text

GroupId – text

FirstName – text

LastName – text

BirthDate - date

Gender – text

Weight – float

3.4.3 Preschool

3.4.3.1 Атрибути

Id – text

Name – text

Description – text

ContactNumber – text

Email – text

City – text

Country – text

Address – text

Website – text

Facilities – text

OpeningHours – datetime

ClosingHours – datetime

OpenRegistration – bool

PricePerDayInUah – decimal

3.4.4 Attendance

3.4.4.1 Атрибути

Id – text

ChildId – text

Date – date

DropOffTime – datetime

PickUpTime – datetime

3.4.5 Application

3.4.5.1 Атрибути

Id – text

ChildId – text

PreschoolId – text

GroupId – text

Status – int

Comment – text

3.4.6 Invoice

3.4.6.1 Атрибути

Id – text

ChildId – text

ExternalId – text

Status – int

DateCreated – datetime

DueDate – datetime

Period – date

PayUrl – text

3.4.7 ScheduleActivity

3.4.7.1 Атрибути

Id – text

ActivityId – text

GroupId – text

StartDateTime – datetime

End DateTime – datetime

3.5 Нефункціональні вимоги

3.5.1 Вимоги до продуктивності

1. Веб-додаток і мобільний клієнт мають відображати вміст у відповідний спосіб, забезпечуючи оптимальну взаємодію з користувачем на різних пристроях і розмірах екрана.

2. Мобільні сторінки мають завантажуватися протягом 5 секунд, зберігаючи стабільну продуктивність веб-версії.

3.5.2 Вимоги до надійності

1. Впроваджувати надійні механізми обробки помилок, щоб ефективно обробляти несподівані помилки та запобігати збоям системи.

2. Вести докладні журнали помилок, включаючи відповідну інформацію для налагодження та аналізу.

3. Забезпечити цілісність транзакцій, особливо в критичних операціях, шляхом впровадження таких механізмів, як транзакції бази даних.

4. Спроектувати систему для плавного відновлення після збою, мінімізуючи вплив на користувачів.

3.5.3 Вимоги до доступності

1. Система має прагнути до мінімум 99,9% часу безвідмовної роботи, враховуючи вікна планового технічного обслуговування.

2. Програма має бути розроблена з механізмами відмовостійкості для обробки несподіваних збоїв у апаратному забезпеченні, програмному забезпеченні або мережевих компонентах.

3. Регулярно виконувати автоматичне резервне копіювання даних програми, включаючи бази даних і конфігураційні файли.

3.5.4 Вимоги до безпеки

1. Система повинна реалізовувати безпечні механізми автентифікації як для веб-програми, так і для мобільного клієнта.

2. Облікові дані користувача мають надійно зберігатися за допомогою галузевих стандартних методів шифрування.

3. Необхідно запровадити контроль доступу на основі ролей, щоб гарантувати, що користувачі мають доступ лише до ресурсів і функцій, які відповідають їхнім ролям.

4. Усі конфіденційні дані, включно з обліковими даними користувача, мають передаватися захищеними каналами з використанням протоколів шифрування, таких як HTTPS.

5. База даних програми повинна реалізовувати шифрування для конфіденційної інформації в стані спокою.

3.5.5 Вимоги до ремонтопридатності

1. Розробити систему з модульною архітектурою, що дозволяє незалежну розробку та підтримку компонентів.

2. Чітко визначені інтерфейси та залежності між модулями для полегшення майбутніх оновлень.

3. Використовуйте систему контролю версій (наприклад, Git) для відстеження змін у вихідному коді, забезпечуючи співпрацю між командами розробників і забезпечуючи історію змін коду.

3.5.6 Вимоги до портативності

1. Веб-програма сумісна з основними веб-браузерами, включаючи, але не обмежуючись ними, Google Chrome, Mozilla Firefox, Microsoft Edge і Safari.

2. Програма працює узгоджено в різних браузерах і версіях.

3. Мобільний клієнт сумісний з основними мобільними операційними системами, такими як iOS і Android.

3.6 Обмеження та винятки (Inverse Requirements)

1. Веб-програма може не працювати оптимально в застарілих або непідтримуваних веб-браузерах.

2. Сумісність не гарантується для операційних систем, відмінних від Windows, macOS і Linux.

3. Мобільний клієнт може не підтримуватися на пристроях із застарілою операційною системою або обмеженими апаратними можливостями.

4. Сумісність обмежена платформами iOS і Android, інші мобільні операційні системи не підтримуються.

5. Хоча мобільний клієнт підтримує основні функції в автономному режимі, певні функції можуть бути обмежені або взагалі не працювати без підключення до Інтернету.

3.7 Архітектурні обмеження

Система має відповідати відповідним галузевим стандартам і вказівкам, таким як стандарти безпеки (наприклад, ISO/IEC 27001) і стандарти веб-доступності (наприклад, WCAG), щоб забезпечити відповідність юридичним і нормативним вимогам.

Система має відповідати положенням про захист даних, таким як Загальний регламент захисту даних (GDPR), забезпечуючи безпечну обробку та обробку даних користувачів.

3.8 Вимоги до збереження даних (Logical Database Requirements)

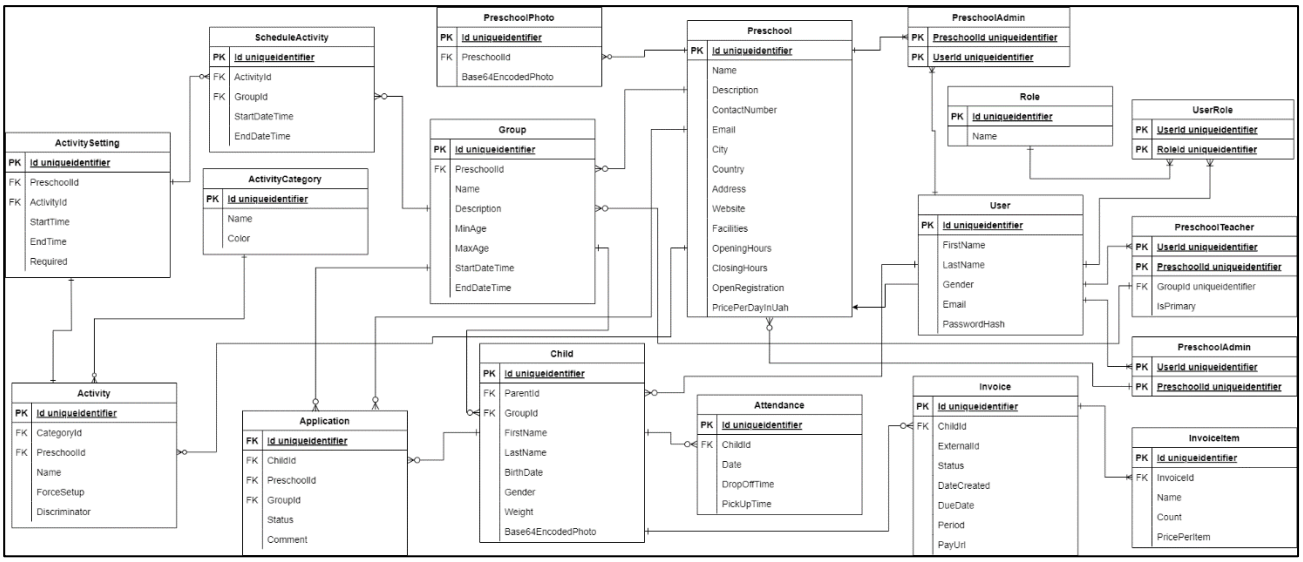


Рисунок 3.5 – ER-діаграма

3.9 Інші вимоги

Інших вимог до продукту немає.

4. Моделі аналізу

4.1 Діаграма послідовності

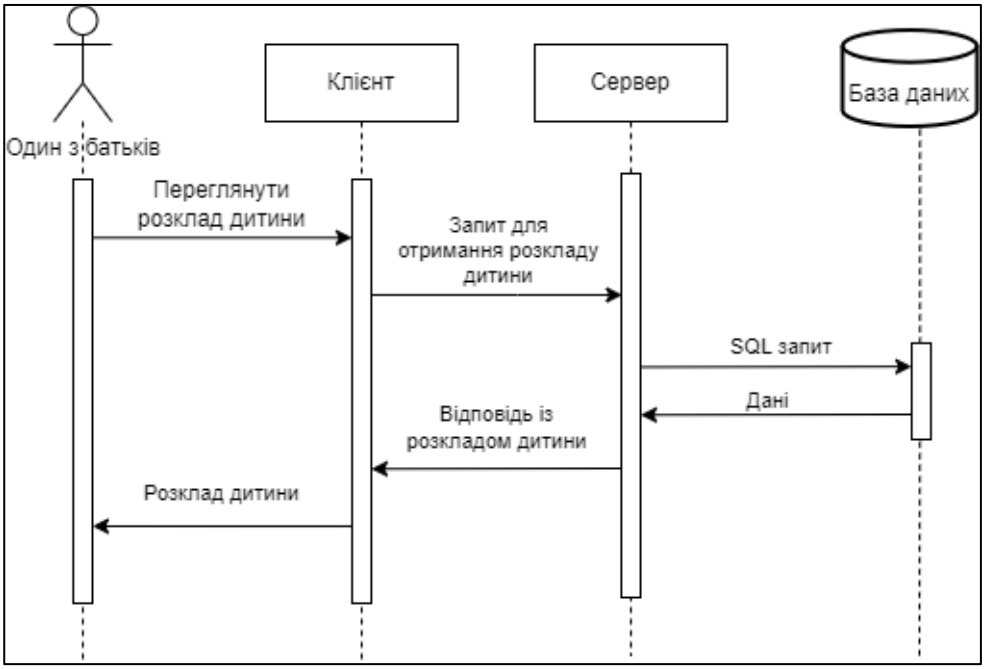


Рисунок 4.1 – Діаграма послідовності перегляду розкладу дитини

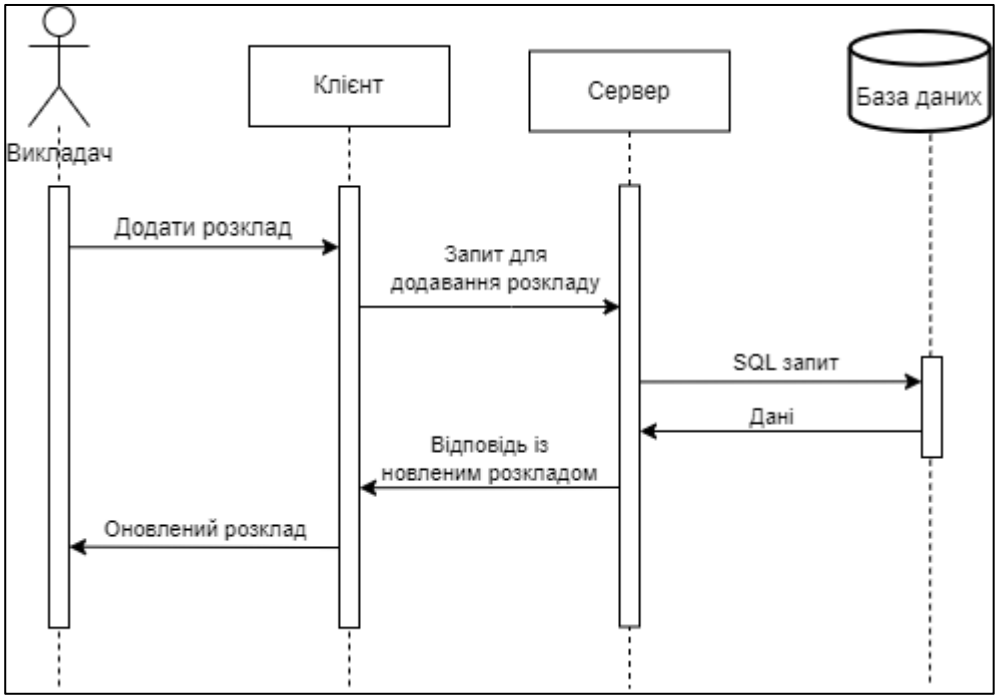


Рисунок 4.2 – Діаграма послідовності додавання розкладу викладачем

4.2 Діаграма переходу станів (STD)

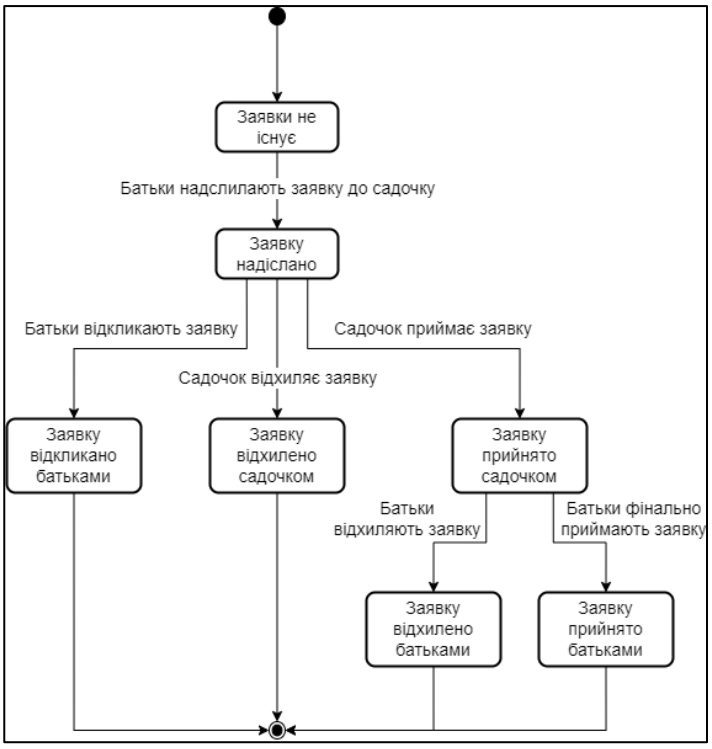


Рисунок 4.3 – Діаграма переходу станів заявки до садочку

Рисунок 4.4 – Діаграма переходу станів відвідуваності дитини

Рисунок 4.5 – Діаграма переходу станів генерації розкладу

4.3 Діаграма потоку даних (DFD)

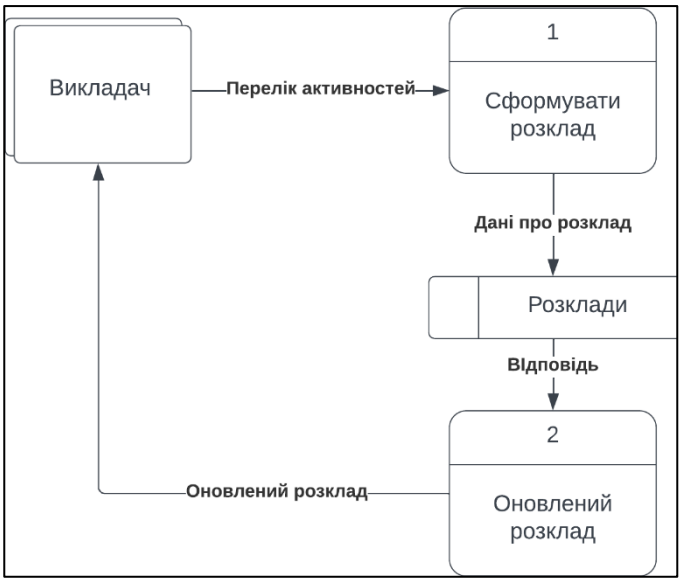


Рисунок 4.6 – Діаграма потоку даних формування розкладу викладачем

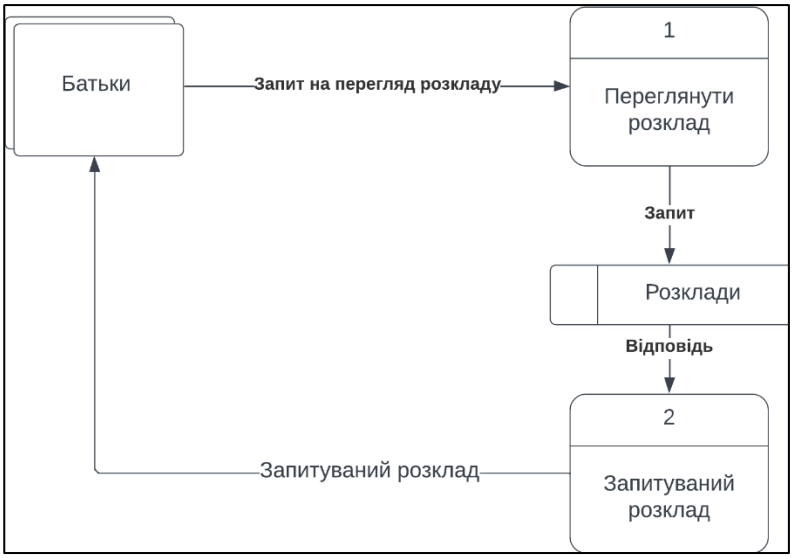


Рисунок 4.7 – Діаграма потоку даних перегляду розкладу батьками

5. Процес управління змінами

Зміни до SRS можуть бути запропоновані будь-яким членом команди, це може бути викликане змінами вимог до системи або обсягу робіт. Наступним кроком повинно бути проведено обговорення щодо змін, після чого визначений член команди вносить схвалені зміни в документ. Оновлена версія має бути внесена в таблицю та підписана кожним членом команди.

ДОДАТОК В

Слайди презентації

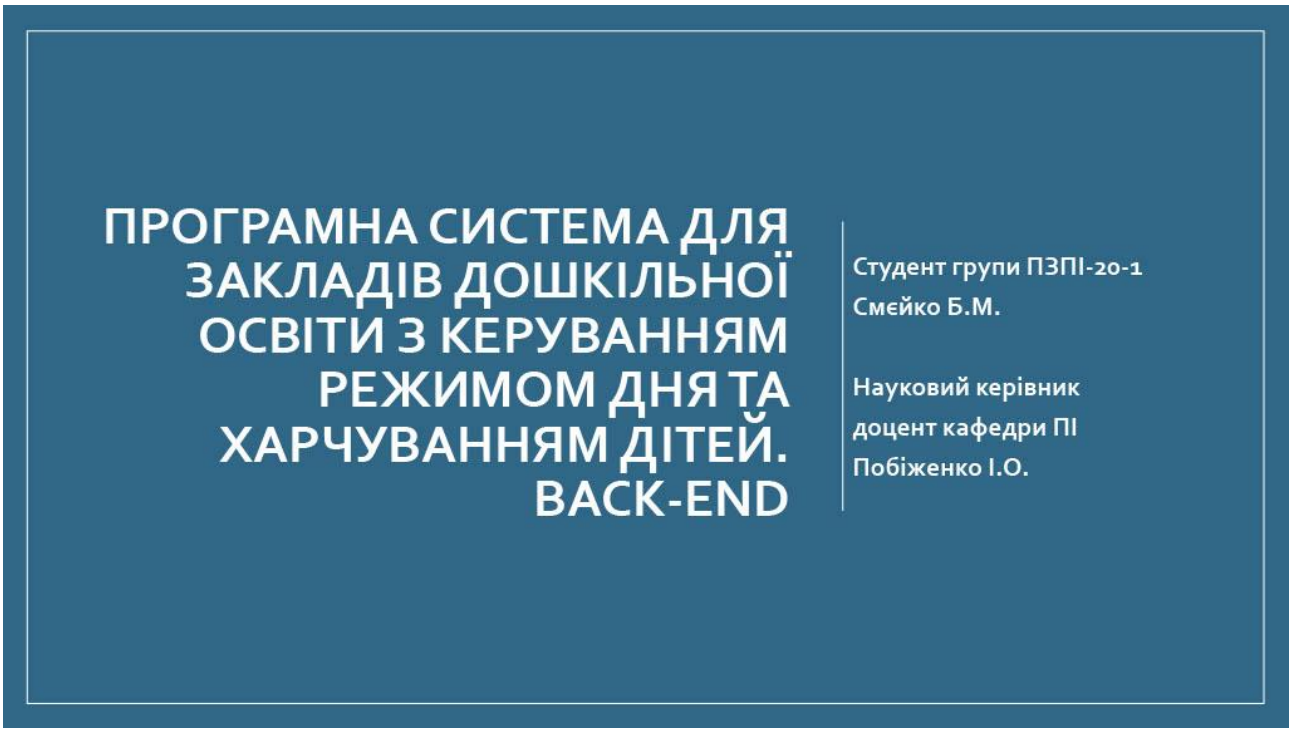


Рисунок В.1 – Слайд презентації №1

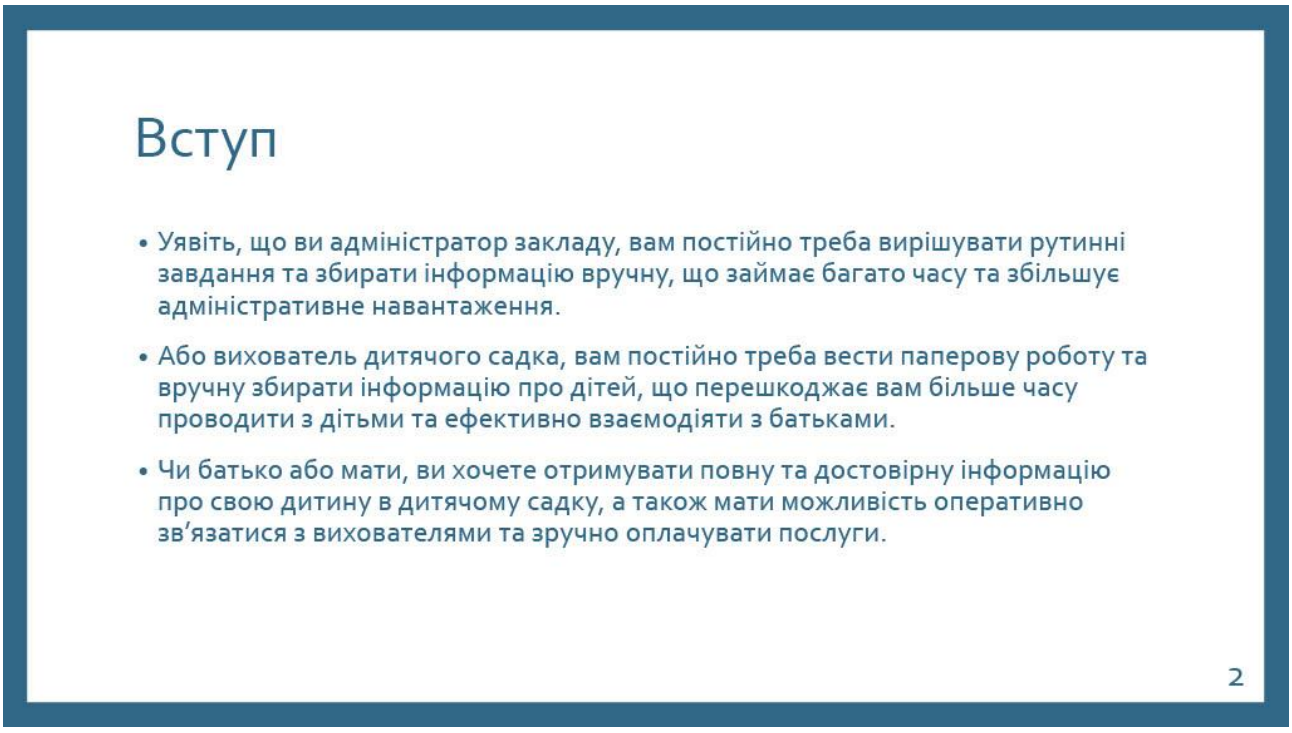


Рисунок В.2 – Слайд презентації №2

Рисунок В.3 – Слайд презентації №3

Функції	KidKare	Brightwheel	EZChildTrack	Procare	Lillio	Програмна система ChildWell
Зручна реєстрація дітей та подання заяв до садків	+/-	+	+	+	+	+
Відстеження присутності та QR-коди	+	+	+	+	+/-	+
Планування режиму дня	-	+	+	+	-	+
Створення власних занять	-	-	-	+	-	+
Автоматичне складання/розподіл розпорядку дня	-	-	-	-	-	+
Рахунки	+	+	+	+	+	+
Адаптованість до ринку України	-	+/-	-	-	+/-	+
Функціонал для батьків	+/-	+	+	+	+	+
Функціонал для вихователів	+	+	+/-	+	+	+
Функціонал для адміністраторів	+	+	+	+	-	+

Рисунок В.4 – Слайд презентації №4

Рисунок В.5 – Слайд презентації №5

Рисунок В.6 – Слайд презентації №6

Рисунок В.7 – Слайд презентації №7

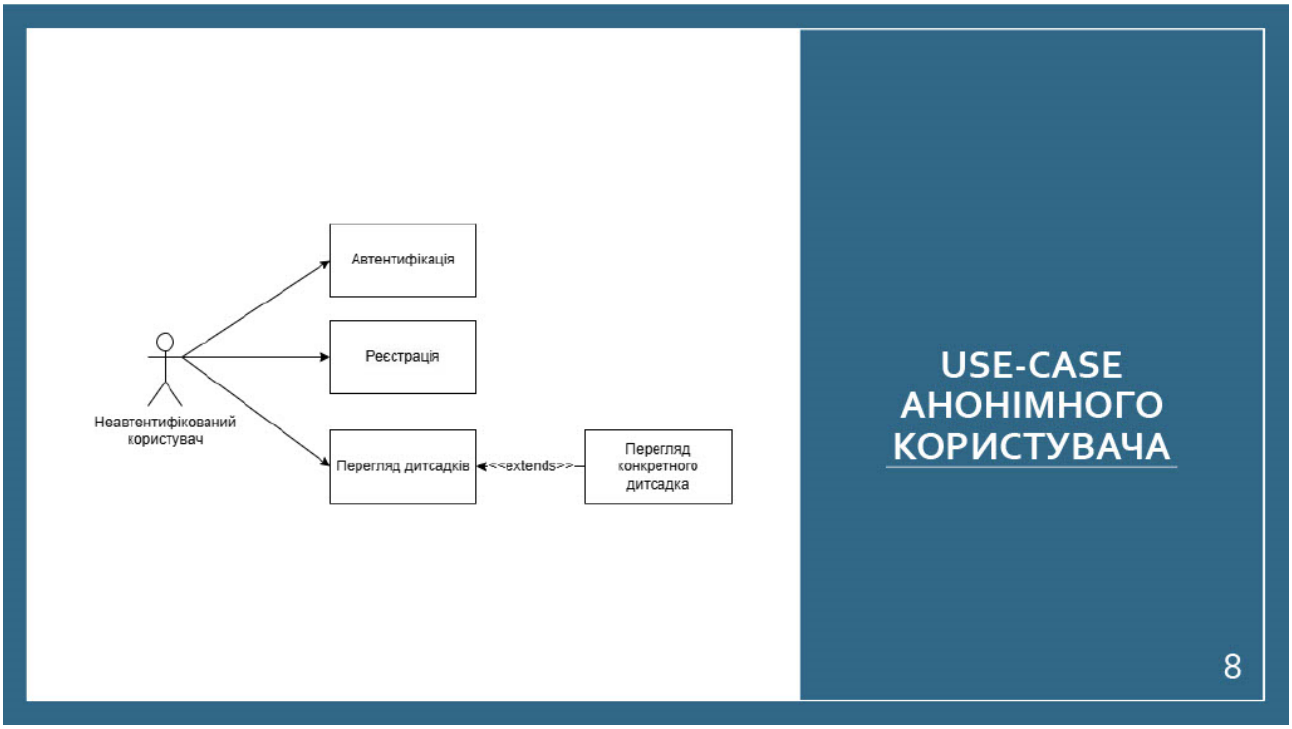


Рисунок В.8 – Слайд презентації №8

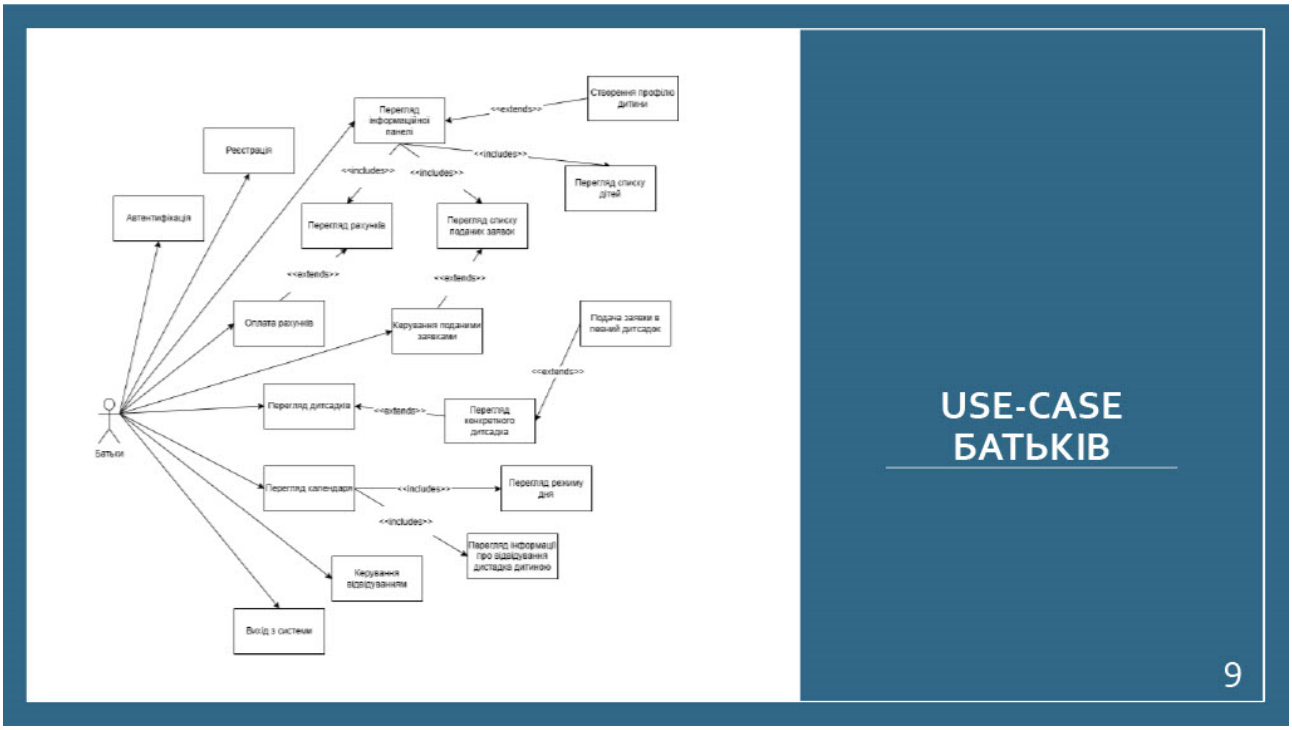


Рисунок В.9 – Слайд презентації №9

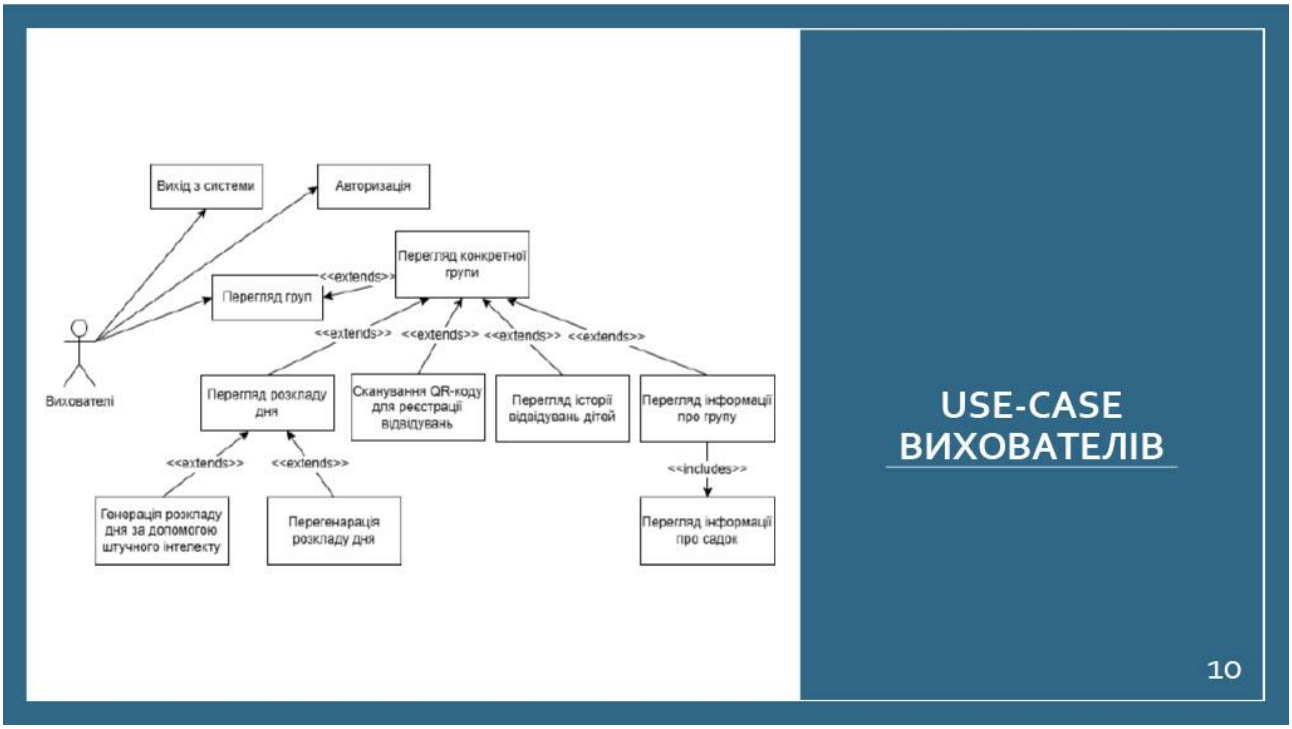


Рисунок В.10 – Слайд презентації №10

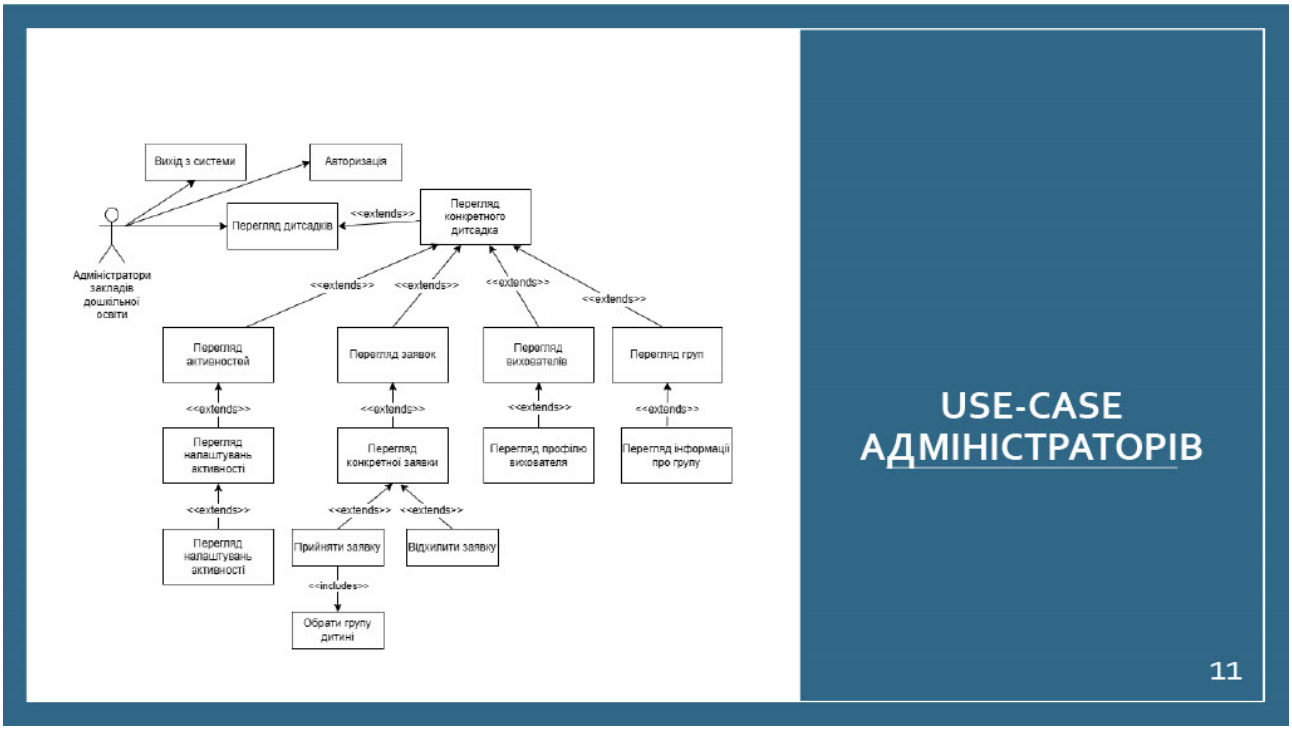


Рисунок В.11 – Слайд презентації №11

Рисунок В.12 – Слайд презентації №12

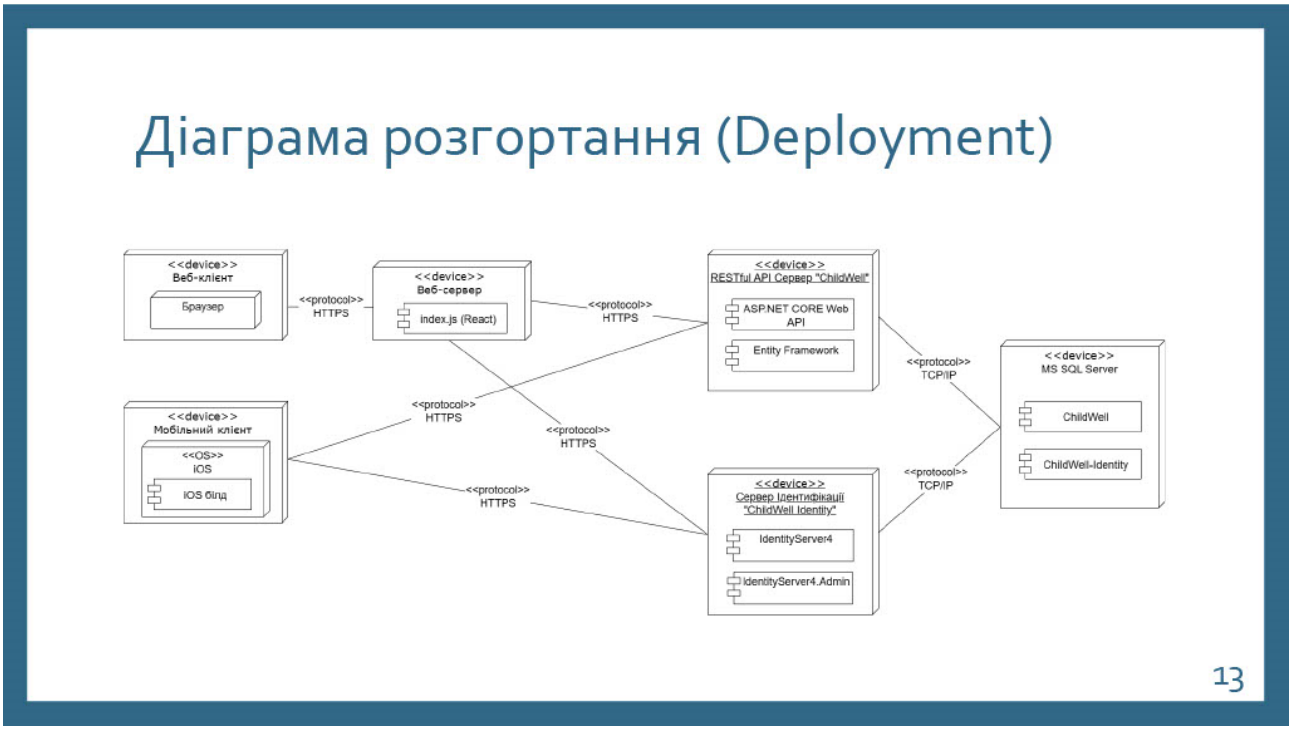


Рисунок В.13 – Слайд презентації №13

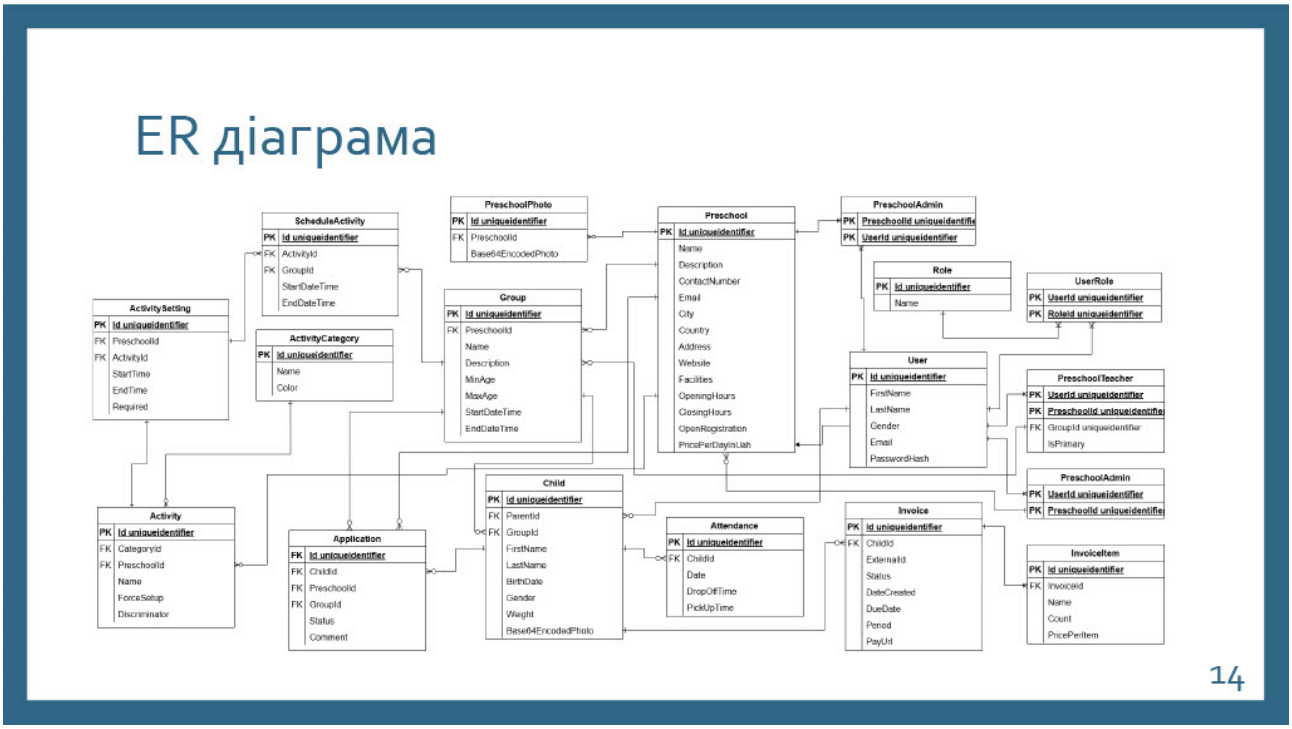


Рисунок В.14 – Слайд презентації №14

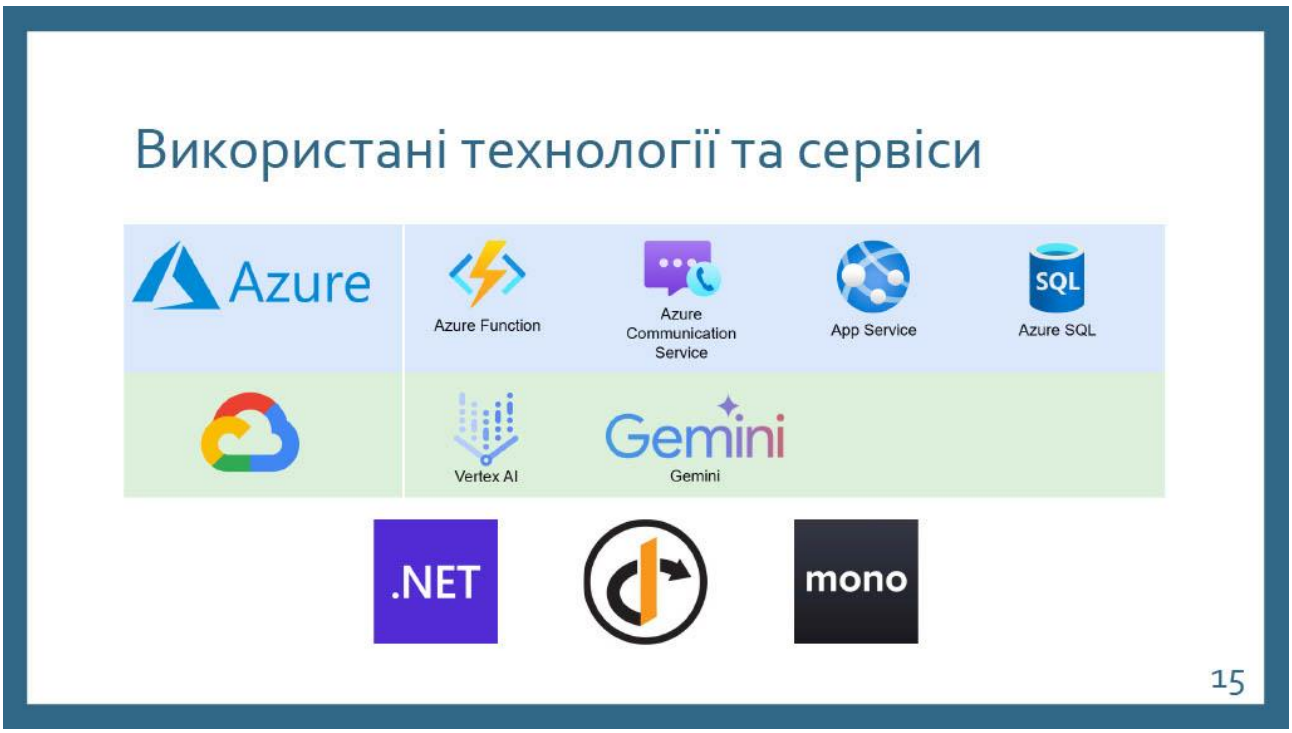


Рисунок В.15 – Слайд презентації №15

Рисунок В.16 – Слайд презентації №16

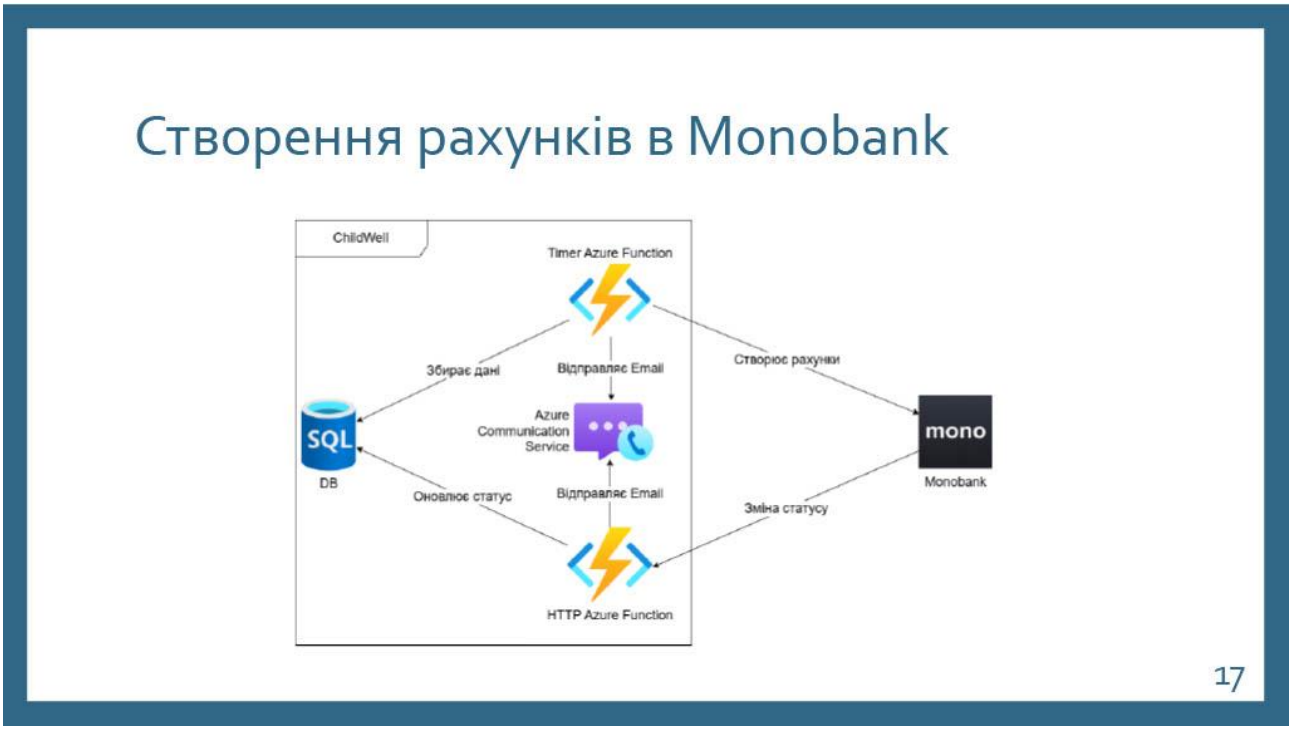


Рисунок В.17 – Слайд презентації №17

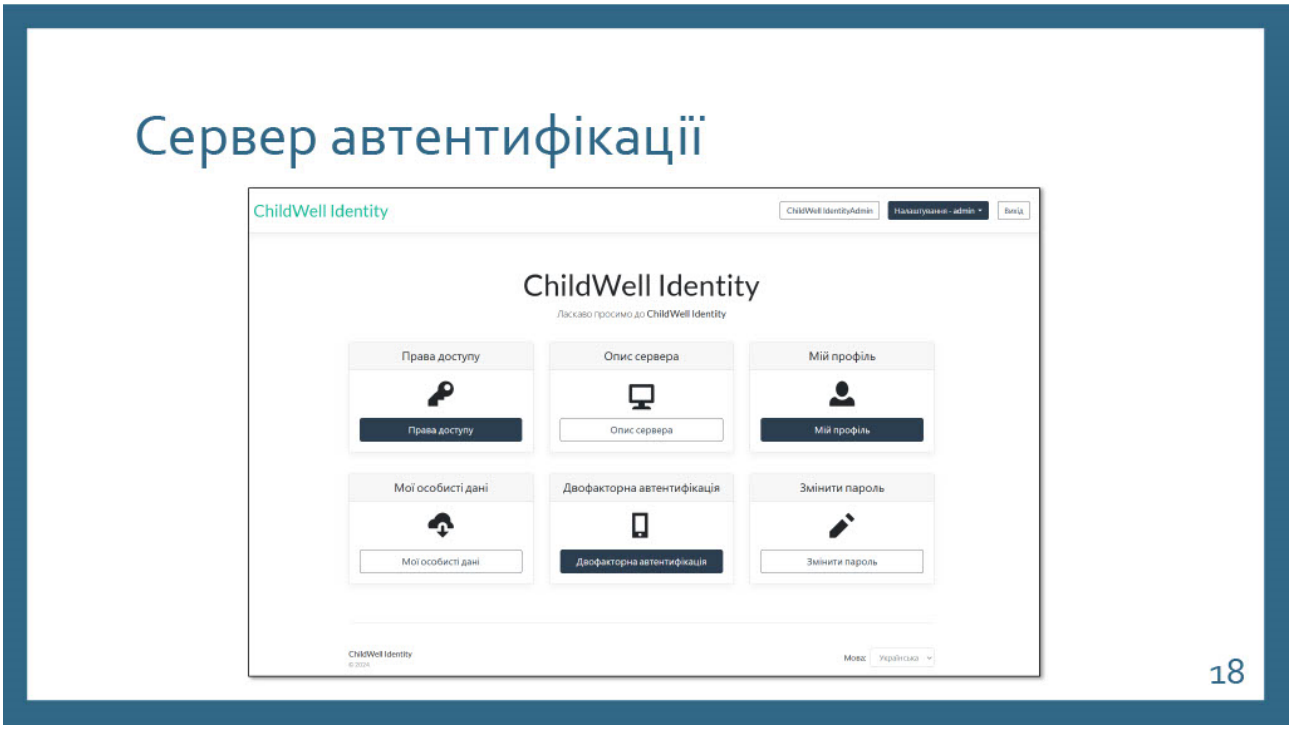


Рисунок В.18 – Слайд презентації №18

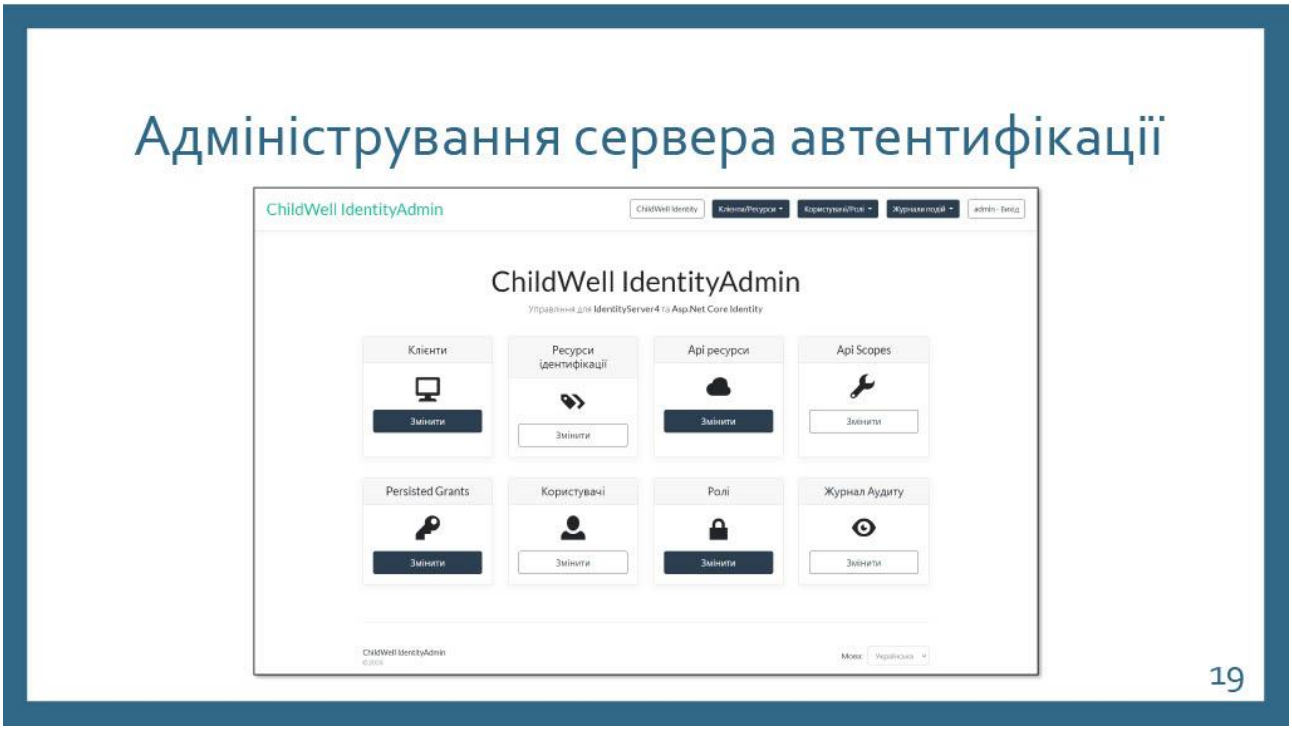


Рисунок В.19 – Слайд презентації №19

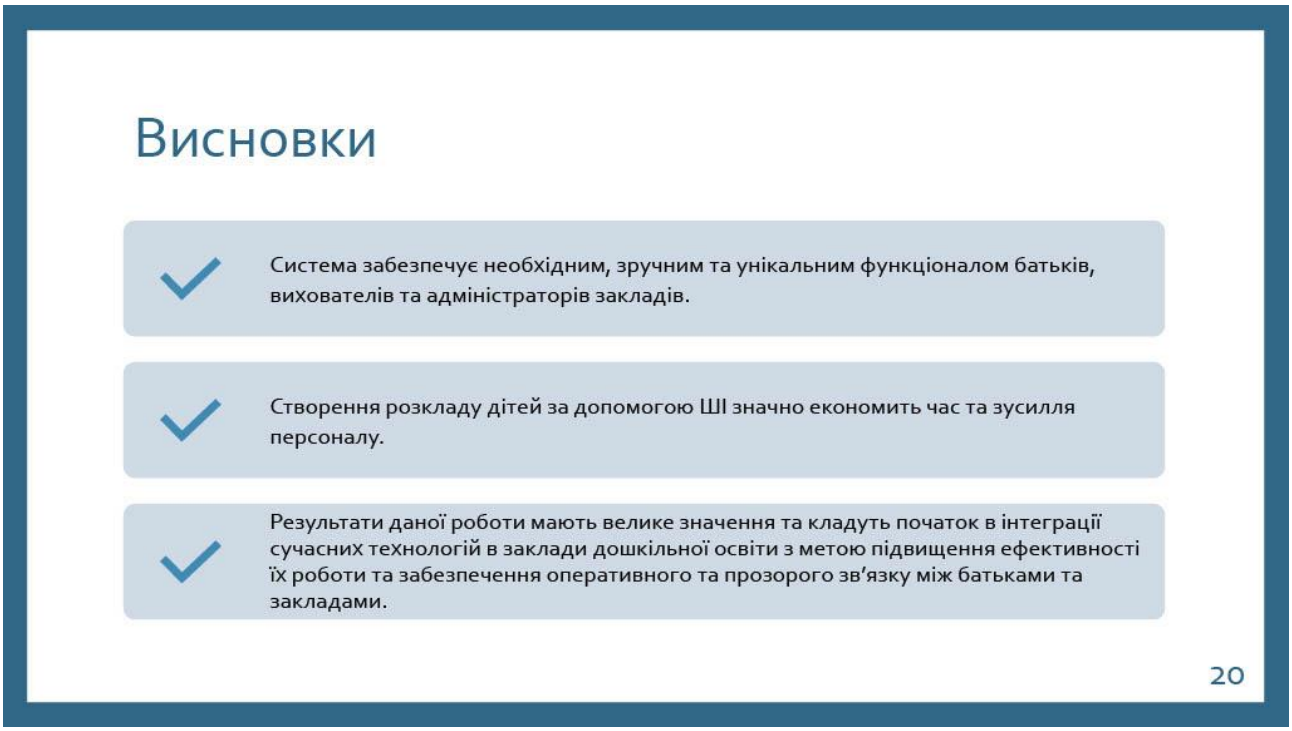


Рисунок В.20 – Слайд презентації №20

ДОДАТОК Г

Специфікація REST API програмної системи

Таблиця Г.1 – Специфікація REST API програмної системи

Метод	**Шлях**	**Тіло запиту**	**Відповіді**
GET	/api/v1/ActivityCategories	Немає	200 (Success): масив ActivityCategoryDto
PATCH	/api/v1/Applications/{id}	UpdateApplicationDto	200 (Success): ApplicationDto, 404 (Not Found): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)
GET	/api/v1/Applications	Немає	200 (Success): масив ApplicationDto, 404 (Not Found): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)

Продовження таблиця Г.1

Метод	**Шлях**	**Тіло запиту**	**Відповіді**
GET	/api/v1/Children	Немає	200 (Success): масив ChildDto, 401 (Unauthorized), 403 (Forbidden)
POST	/api/v1/Children	ChildCreateDto	201 (Created): ChildDto, 401 (Unauthorized), 403 (Forbidden)
GET	/api/v1/Children/{id}	Немає	200 (Success): ChildDto, 404 (Not Found): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)
GET	/api/v1/Children/{id}/attendance	Немає	200 (Success): масив ChildAttendanceEntry, 404 (Not Found): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)

Продовження таблиця Г.1

Метод	**Шлях**	**Тіло запиту**	**Відповіді**
GET	/api/v1/Children/{id}/attendance/qr-code-data	Немає	200 (Success): string, 404 (Not Found): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)
POST	/api/v1/Children/attendance/mark	EncryptedQrCodeDataDto	200 (Success): AttendanceMarkedDto, 401 (Unauthorized), 403 (Forbidden)
GET	/api/v1/Invoices	Немає	200 (Success): масив InvoiceDto, 401 (Unauthorized), 403 (Forbidden)

Продовження таблиця Г.1

Метод	**Шлях**	**Тіло запиту**	**Відповіді**
GET	/api/v1/Preschools/{preschoolId}/groups	Немає	200 (Success): масив GroupDto, 404 (Not Found): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)
POST	/api/v1/Preschools/{preschoolId}/groups	GroupCreateDto	201 (Created): масив GroupDto, 404 (Not Found): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)
GET	/api/v1/Preschools/{preschoolId}/groups/{groupId}	Немає	200 (Success): GroupDto, 404 (Not Found): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)

Продовження таблиця Г.1

Метод	Шлях	Тіло запиту	Відповіді
PATCH	/api/v1/Preschools/{preschoolId}/groups/{groupId}	GroupCreateDto	200 (Success): GroupDto, 404 (Not Found): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)
DELETE	/api/v1/Preschools/{preschoolId}/groups/{groupId}	Немає	204 (No Content), 404 (Not Found): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)
POST	/api/v1/Preschools/{preschoolId}/groups/{groupId} /teachers	Немає	200 (Success), 404 (Not Found): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)
GET	/api/v1/Preschools/{preschoolId}/groups/{groupId} /children	Немає	200 (Success): масив ChildDto, 404 (Not Found): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)

Продовження таблиця Г.1

Метод	Шлях	Тіло запиту	Відповіді
GET	/api/v1/Preschools/{preschoolId}/groups/{groupId}/attendance	Немає	200 (Success): масив GroupAttendanceEntry, 404 (Not Found): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)
POST	/api/v1/Preschools/{preschoolId}/groups/{groupId}/schedule/export-to-google	Немає	200 (Success), 404 (Not Found): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)
GET	/api/v1/Preschools/{preschoolId}/groups/{groupId}/schedule	Немає	200 (Success): масив ScheduleActivityDto, 404 (Not Found): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)

Продовження таблиця Г.1

Метод	Шлях	Тіло запиту	Відповіді
PATCH	/api/v1/Preschools/{preschoolId}/groups/{groupId}/schedule	масив ScheduleActivityDto	200 (Success), 404 (Not Found): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)
POST	/api/v1/Preschools/{preschoolId}/groups/{groupId}/schedule/preview	CreateScheduleRequest	200 (Success): масив ScheduleActivityDto, 400 (Bad Request): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)
POST	/api/v1/Preschools/{preschoolId}/groups/{groupId}/schedule/finalize	Немає	204 (No Content), 400 (Bad Request): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)
POST	/api/v1/Preschools/{preschoolId}/admins	масив string (uuid)	200 (Success), 404 (Not Found): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)

Продовження таблиця Г.1

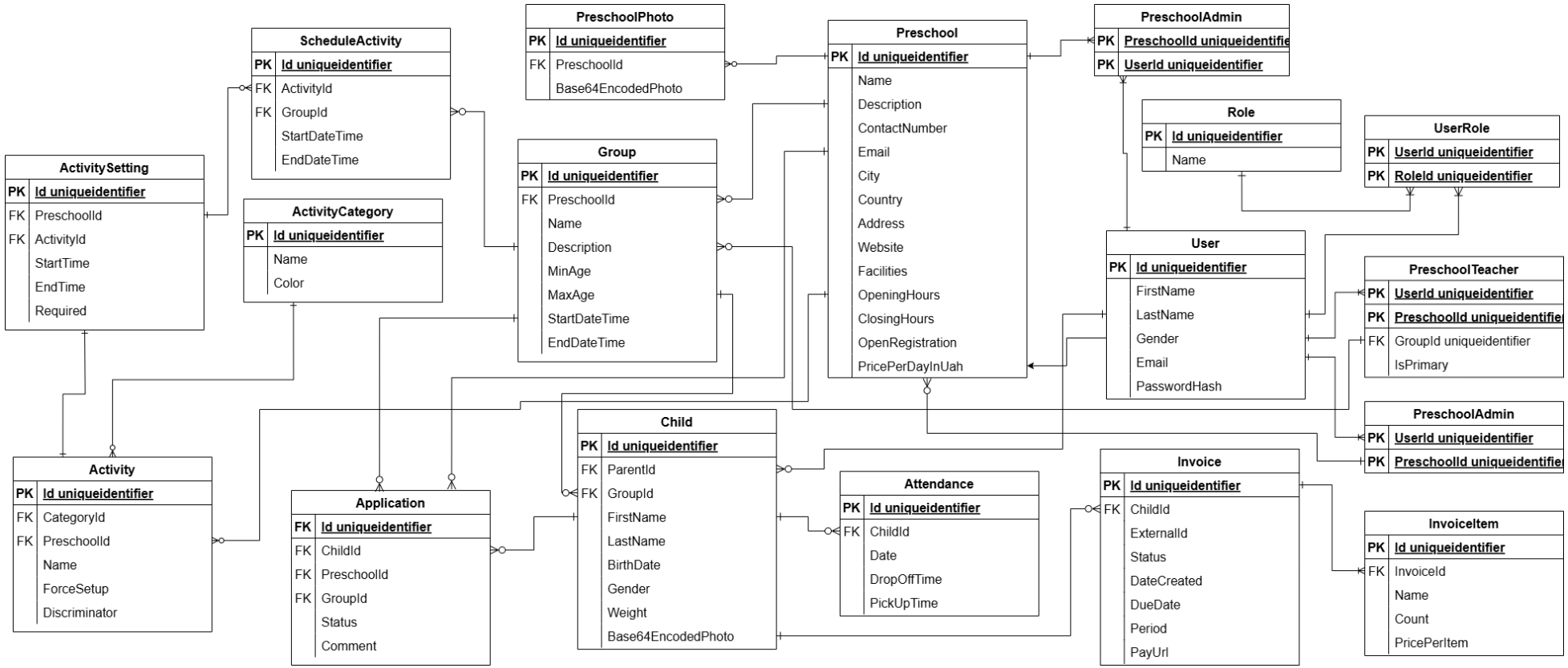
Метод	Шлях	Тіло запиту	Відповіді
GET	/api/v1/Preschools/{preschoolId}/teachers	Немає	200 (Success): масив TeacherDto, 404 (Not Found): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)
POST	/api/v1/Preschools/{preschoolId}/teachers	масив string (uuid)	200 (Success), 404 (Not Found): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)
GET	/api/v1/Preschools/{preschoolId}/applications	Немає	200 (Success): масив ApplicationDto, 404 (Not Found): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)
GET	/api/v1/Preschools/{preschoolId}/activities	Немає	200 (Success): масив ActivityDto, 404 (Not Found): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)
GET	/api/v1/Preschools/{preschoolId}/activities/settings	Немає	200 (Success): масив ActivitySettingDto, 401 (Unauthorized), 403 (Forbidden)

Кінець таблиця Г.1

Метод	Шлях	Тіло запиту	Відповіді
PATCH	/api/v1/Preschools/{preschoolId}/activities/settings	масив ActivitySettingDto	204 (No Content), 404 (Not Found): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)
POST	/api/v1/Preschools/{preschoolId}/apply	масив ApplyToPreschoolDto	200 (Success), 404 (Not Found): ProblemDetails, 409 (Conflict): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)
GET	/api/v1/Preschools/public	Немає	200 (Success): масив PreschoolDto
GET	/api/v1/Preschools/public/{preschoolId}	Немає	200 (Success): PreschoolDto, 404 (Not Found): ProblemDetails
GET	/api/v1/TeacherGroups	Немає	200 (Success): GroupDto, 404 (Not Found): ProblemDetails, 401 (Unauthorized), 403 (Forbidden)

ДОДАТОК Д

Діаграма сутностей (Entity-Relationship) системи



ДОДАТОК Е

Код функції (Azure Function) зворотнього виклику (webhook) для оновлення статусу рахунків

```
[Function(nameof(InvoiceStatusUpdateFunction))]
public async Task<HttpResponseData> RunAsync(
    [HttpTrigger(AuthorizationLevel.Anonymous, "post")]
    HttpRequestData req
)
{
    try
    {
        var stringBody = await req
            .ReadAsStringAsync() ?? string.Empty;
        var monoStatus = JsonConvert
         .DeserializeObject<MonoInvoiceStatusResponse>(stringBody);

        await _invoiceService
            .UpdateInvoiceStatus(
                monoStatus.InvoiceId,
                monoStatus.Status,
                monoStatus.ModifiedDate
            );

        var res = req.CreateResponse(HttpStatusCode.OK);
        res.Headers.Add("X-Sign", CreateSignature(stringBody));
        return res;
    }
    catch (Exception e)
    {
        _logger.LogError($"Error occured: {e}");
    }

    return req.CreateResponse(HttpStatusCode.InternalServerError);
}
```

ДОДАТОК Ж

Код класу GoogleChatSession, що відповідальний за взаємодію з Google Vertex AI

```
public class GoogleChatSession : IAIChatSession
{
    private readonly string _modelPath;
    private readonly PredictionServiceClient
                                        _predictionServiceClient;
    private readonly List<Content> _contents;

    private readonly ILogger<GoogleChatSession> _logger;

    public GoogleChatSession(
        ILogger<GoogleChatSession> logger,
        IOptions<GoogleVertexAIConfiguration> vertexAiOptions
    )
    {
        var vertexAi = vertexAiOptions.Value;
        _logger = logger;
        _modelPath = $"projects/{vertexAi.ProjectId}/" +
                     $"locations/{vertexAi.Location}/" +
                     $"publishers/{vertexAi.Publisher}/" +
                     $"models/{vertexAi.Model}";
        _predictionServiceClient =
            new PredictionServiceClientBuilder
            {
                CredentialsPath = vertexAi.CredentialsPath,
                Endpoint =
                    $"{vertexAi.Location}" +
                    $"-aiplatform.googleapis.com",
                Logger = logger
            }.Build();
        _contents = new List<Content>();
    }

    public async Task<string> SendMessageAsync(
        string prompt,
        string? sessionKey = null
    )
    {
        if (
            !string.IsNullOrEmpty(sessionKey) &&
            ChatSessionStore<GoogleChatSession>
                .Get(sessionKey) != null
        )
        {
            _contents.Clear();
            _contents
                .AddRange(
                    ChatSessionStore<GoogleChatSession>
                    .Get(sessionKey)
                    ._contents
                );
        }
```

```
            var content = new Content
            {
                Role = "USER",
                Parts =
                {
                    new List<Part>()
                    {
                        new()
                        {
                            Text = prompt
                        }
                    }
                }
            };
            _contents.Add(content);

            var generateContentRequest = new GenerateContentRequest
            {
                Model = _modelPath,
                Contents = { _contents },
                GenerationConfig = new GenerationConfig
                {
                    Temperature = 0.2f,
                    TopP = 1,
                    TopK = 32,
                    CandidateCount = 1,
                    MaxOutputTokens = 8192
                }
            };

            var response = await _predictionServiceClient
                .GenerateContentAsync(generateContentRequest);

            _contents.Add(response.Candidates[0].Content);

            if (!string.IsNullOrEmpty(sessionKey))
            {
                ChatSessionStore<GoogleChatSession>
                    .Add(sessionKey, this);
            }

            return response.Candidates[0].Content.Parts[0].Text;
        }

        public void ClearSession(string sessionKey)
        {
            ChatSessionStore<GoogleChatSession>
                .Remove(sessionKey);
        }
    }
```

ДОДАТОК И

Код класу AzureCommunicationEmailSender, що відповідальний за відправку повідомлень на електронну пошту за допомогою Azure Communication Service

```
public class AzureCommunicationEmailSender : IEmailSender
{
    private readonly
        ILogger<AzureCommunicationEmailSender> _logger;
    private readonly
        AzureCommunicationConfiguration _configuration;
    private readonly EmailClient _emailClient;

    public AzureCommunicationEmailSender(
        ILogger<AzureCommunicationEmailSender> logger,
        AzureCommunicationConfiguration configuration
    )
    {
        _logger = logger;
        _configuration = configuration;
        _emailClient =
            new EmailClient(_configuration.ConnectionString);
    }

    public Task SendEmailAsync(
        string email, string subject, string htmlMessage
    )
    {
        _logger.LogInformation(
                $"Sending email: {email}, " +
                $"subject: {subject}, " +
                $"message: {htmlMessage}");
        try
        {
            EmailSendOperation emailSendOperation =
                _emailClient.Send(
                    WaitUntil.Completed,
                    senderAddress: _configuration.Sender,
                    recipientAddress: email,
                    subject: subject,
                    htmlContent: htmlMessage);
            _logger.LogInformation(
                $"Email: {email}, " +
                $"subject: {subject}, " +
                $"message: {htmlMessage} successfully sent");
            return Task.CompletedTask;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"Exception {ex} during sending email: {email}, " +
                $"subject: {subject}");
            throw;
        }
    }
}
```

ДОДАТОК К

Диплом ІІ ступеня в рамках 28-го Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті»

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

ДИПЛОМ

НАГОРОДЖУЄТЬСЯ

авторський колектив у складі

Бондаренко А.А., Нестеренко В.В., Смєйко Б.М.

(групи ПЗПІ-20-1, ПЗПІ-20-3,
Харківський національний університет радіоелектроніки),

який здобув **ІІ** місце

на секції

«Програмна Інженерія. Інформаційні технології в освіті»

конференції

«Інформаційні інтелектуальні системи»

в рамках **28-го** Міжнародного молодіжного форуму

«РАДІОЕЛЕКТРОНІКА І МОЛОДЬ В XXI СТОЛІТТІ»

з доповіддю

«Аналіз використання штучного інтелекту для генерації розкладу занять і харчування дітей в закладах дошкільної освіти»

Голова конференції «Інформаційні інтелектуальні системи», д.т.н., проф. Андрій ЄРОХІН

м. Харків, 2024