**ЗМІСТ**

[ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ 5](#_Toc11746347)

[ВСТУП 7](#_Toc11746348)

[1. ОГЛЯД ІСНУЮЧИХ СИСТЕМ-АНАЛОГІВ 10](#_Toc11746349)

[1.1 Класифікація агрегаторів 10](#_Toc11746350)

[1.2. Короткий огляд систем-аналогів 12](#_Toc11746351)

[2. АНАЛІЗ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ПРОЕКТУ 19](#_Toc11746352)

[2.1 Основні особливості розробки інтерфейсів 19](#_Toc11746353)

[2.1.1 Інтуїтивно зрозумілий інтерфейс 19](#_Toc11746354)

[2.1.2 Самодокументований інтерфейс 20](#_Toc11746355)

[2.1.3 Ненав’язливий інтерфейс 20](#_Toc11746356)

[2.1.4 Перевірка даних 21](#_Toc11746357)

[2.1.5 Мінімізація помилок користувача 21](#_Toc11746358)

[2.2 Проектування інтерфейсу 22](#_Toc11746359)

[2.3 Технології розробки інтерфейсів 22](#_Toc11746360)

[2.4 Технології розробки бізнес-аналітики 25](#_Toc11746361)

[2.5 Технології монетизації 31](#_Toc11746362)

[3. РЕАЛІЗАЦІЯ ОБРАНИХ МОДУЛІВ СИСТЕМИ 34](#_Toc11746363)

[3.1 Реалізація інтерфейсу 34](#_Toc11746364)

[3.2 Реалізація модуля монетизації 40](#_Toc11746365)

[3.2.1 Інтеграція реклами 40](#_Toc11746366)

[3.2.2 Система стягування комісії 42](#_Toc11746367)

[3.3 Реалізація модуля аналітики 47](#_Toc11746368)

[4. ОПИС ІНТЕРФЕЙСУ ТА ІНСТРУКЦІЇ КОРИСТУВАННЯ 50](#_Toc11746369)

[4.1 Робота з бібліотекою елементів для інтерфейсу 50](#_Toc11746370)

[4.2 Робота з модулем монетизації 51](#_Toc11746371)

[4.3 Робота з модулем аналітики 51](#_Toc11746372)

[5. ЕКОНОМІЧНА ЧАСТИНА 53](#_Toc11746373)

[5. ВИСНОВОК 67](#_Toc11746374)

[СПИСОК ЛІТЕРАТУРИ 68](#_Toc11746375)

# **ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ**

API (Application programming interface) – прикладний програмний інтерфейс.

XML (Extensible Markup Language). – розширювана мова розмітки.

HTML (Hypertext Markup Language) - стандартна [мова розмітки](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D1%80%D0%BE%D0%B7%D0%BC%D1%96%D1%82%D0%BA%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) для створення [веб-сторінок](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D0%BE%D1%80%D1%96%D0%BD%D0%BA%D0%B0) і [веб-додатків](https://uk.wikipedia.org/w/index.php?title=%D0%92%D0%B5%D0%B1-%D0%B4%D0%BE%D0%B4%D0%B0%D1%82%D0%BA%D1%96%D0%B2&action=edit&redlink=1).

SQL (Structured query language — мова структурованих запитів) - декларативна мова програмування для взаємодії користувача з базами даних.

MySQL – система керування реляційними базами даних з відкритим кодом.

HTTP (Hyper Text Transfer Protocol) – протокол передачі даних, що використовується в комп'ютерних мережах.

HTTPS - HTTP, який передається через SSL або TLS.

SSL (Secure Sockets Layer — рівень захищених сокетів) — криптографічний протокол, який забезпечує безпечне з'єднання між клієнтом і сервером.

TLS (Transport Layer Security) — захист на транспортному рівні) — криптографічний протокол, що надає можливості безпечної передачі даних в Інтернет для навігації, отримання пошти, спілкування, обміну файлами, тощо. Використовує асиметричне шифрування.

Брейкпойнт (breakpoint) – значення ширини екрану в пікселях, на якій відбуваються зміни в CSS-стилях фреймворку Bootstrap.

Фреймворк - інфраструктура програмних рішень, що полегшує розробку складних систем.

Еквайр — банк або інша фінансова установа, що надає послуги еквайрингу, тобто, здійснює розрахунки з підприємствами, які приймають оплату від власників платіжних карток за товари чи послуги або видають їм готівку.

Агрегатор - програма або сервіс, яка збирає дані з різних джерел в одному місці.

Монетизація – отримання прибутку з веб-сервісів.

UI (User Interface) - засіб зручної взаємодії користувача з інформаційною системою.

UX (User eXperience) - це те, що людина відчуває при користуванні продуктом, системою чи сервісом.

Адаптивність інтерфейсу – властивість інтерфейсу підлаштовуватись та виглядати однаково добре на пристроях із різними розмірами екранів.

# **ВСТУП**

Останні роки спостерігається стрімке зростання кількості товарних агрегаторів та сервіс-агрегаторів. Такі платформи вигідні для всіх категорій користувачів – споживач отримує багато пропозицій в одному місці і може швидко порівнювати та обирати їх, а продавець має місце для розміщення цих пропозицій, може легко моніторити ціни конкурентів та отримувати детальну аналітику ринку. Власник, забезпечуючи тільки розробку, підтримку та просування системи отримує значні доходи.

Саме ці особливості роблять агрегатори все популярнішими. При цьому існують спеціалізовані агрегатори, які направлені на обслуговування, наприклад, служб таксі, і в таких випадках споживач не може самостійно обирати постачальника послуг – він вибирається автоматизовано для опимізації розподілу замовлень

Для користувачів агрегаторів дуже важливим є **користувацький інтерфейс**, оскільки саме через нього відбувається взаємодіють із системою. Інтерфейс часто виявляється тим чинником, через який можуть суттєво мінятися конверсії системи, оскільки неякісні рішення викликають проблеми з навігацією, і відповідно втрату потенційних користувачів, що можуть у майбутньому принести дохід.

Зі сторони бізнесу, який використовує агрегатор як торгову платформу, та з точки зору власників агрегаторів, важлива детальна **бізнес-аналітика** для оцінки власної ефективності та кращого розуміння своїх бізнес-процесів. Все це дозволяє швидко реагувати на зміни ситуації на ринку та виявляти проблемні місця, що спрямовано на постійне вдосконалення та підвищення доходів. Аналітика повинна відображати всі основні показники ефективності та надавати основу для розуміння позитивних і негативних чинників впливу на ведення справ.

Окрім того, для власників агрегаторів найважливішою частиною системи є **монетизація**, адже, зазвичай, розробка, підтримка та просування агрегатора скеровані на отримання доходів його власником чи інвестором. Монетизація обов’язково повинна бути автоматизованою, тобто такою що не вимагає жодних додаткових дій від користувача, бізнесу чи власника. При цьому важливо витримувати баланс між вигодою для власника та додатковими витратами для користувача і бізнесу.

Метою бакалаврської кваліфікаційної роботи є реалізація інтерфейсу, бізнес-аналітики та монетизації для системи автоматизованого розподілу замовлень. Кожна категорія людей має різні головні потреби, і щоб якнайповніше їх задовільнити, розробляючи агрегатор критично необхідно реалізувати дружній до користувача інтерфейс, детальну аналітику для бізнесу та вигідну для власника і водночас добре збалансовану монетизацію.

# **1. ОГЛЯД ІСНУЮЧИХ СИСТЕМ-АНАЛОГІВ**

## **1.1 Класифікація агрегаторів**

На даний момент існує досить багато агрегаторів. Їх всіх можна умовно поділити на кілька категорій. В залежності від об’єкта агрегації поділ можна реалізувати таким чином:

А) Товарні агрегатори передставляють собою платформи які об’єднують дані з багатьох інтернет-магазинів і надають інформацію про всі їхні товари в одному місці. Якщо вважати що товарний агрегатор це продуктовий ринок, то інтернет-магазин буде яткою. Окрім інтернет-магазинів, одиницями товарних агрегаторів можуть бути звичайні продавці, якщо в системі передбачено таку можливість. Яскравим прикладом товарного агрегатора можуть бути такі сервіси як Hotline та Ekatalog.

Б) Сервіс-агрегатори – більш вузькоспрямовані аналоги товарних агрегаторів, які представляють вже не товари, а послуги, зазвичай, в межах однієї сфери (наприклад, доставка їжі, замовлення таксі, каршерінг тощо). В таких системах розподіл замовлень, якщо він потрібен, найчастіше відбувається автоматизовано. Приклади таких агрегаторів – Glovo, Grabr, Uklon тощо. В цю категорію входять також платіжні агрегатори.

В) Інформаційні агрегатори – портали новин або статей, які колекціонують посилання на контент з багатьох різних ЗМІ, блогів та інших інформаційних сатйів. Найкращим прикладом є Google Feed – стрічка новин, в якій користувач може самостійно регулювати частоту видачі контенту з кожного окремого джерела. Інформаційні агрегатори монетизуються виключно завдяки інтеграції рекламних оголошень.

Г) Агрегатори змішаного типу – надають збірки даних кількох типів водночас. Є найбільш рідкісним типом, і за своєю суттю є агрегаторами лише частково. Найчастіше для користувача це виглядає радше як дошка оголошень. В більшості випадків мають можливість розміщення пропозицій як бізнесом, так і фізичними особами. Найкращий приклад – загальновідомий OLX, який діє не тільки в Україні, а й у Румунії, Болгарії, Казахстані, Білорусі, Угорщині, Бразилії та Польщі.

Окрім об’єкту агрегації, важливим фактором класифікації агрегаторів є метод розподілу замовлень, у випадках коли він потрібен (наприклад, в інформаційних агрегаторах потреби в розподілі замовлень немає через очевидні причини) . Тут можна виділити три основні типи:

А) Агрегатори з ручним розподілом замовлень. В системах такого типу користувач самостійно вибирає продавця чи надавача послуг. Найпростіший з технічної точки зору вид, оскільки потрібно лише надати користувачу список пропозицій, а всі наступні кроки для покупки товару чи послуги він виконає сам. Використовується, зазвичай, на дошках оголошень та товарних агрегаторах.

Б) Агрегатори з автоматизованим розподілом замовлень. Для даного типу притаманна відсутність можливості для користувача самостійно обирати продавця чи надавача послуг. Система робить все автоматизовано, грунтуючись на вхідних даних та обираючи найоптимальніший варіант для поточного набору вхідних даних. Іноді в таких системах використовується штучний інтелект. Найчастіше автоматизований розподіл замовлень використовують у вузьконаправлених сервіс-агрегаторах.

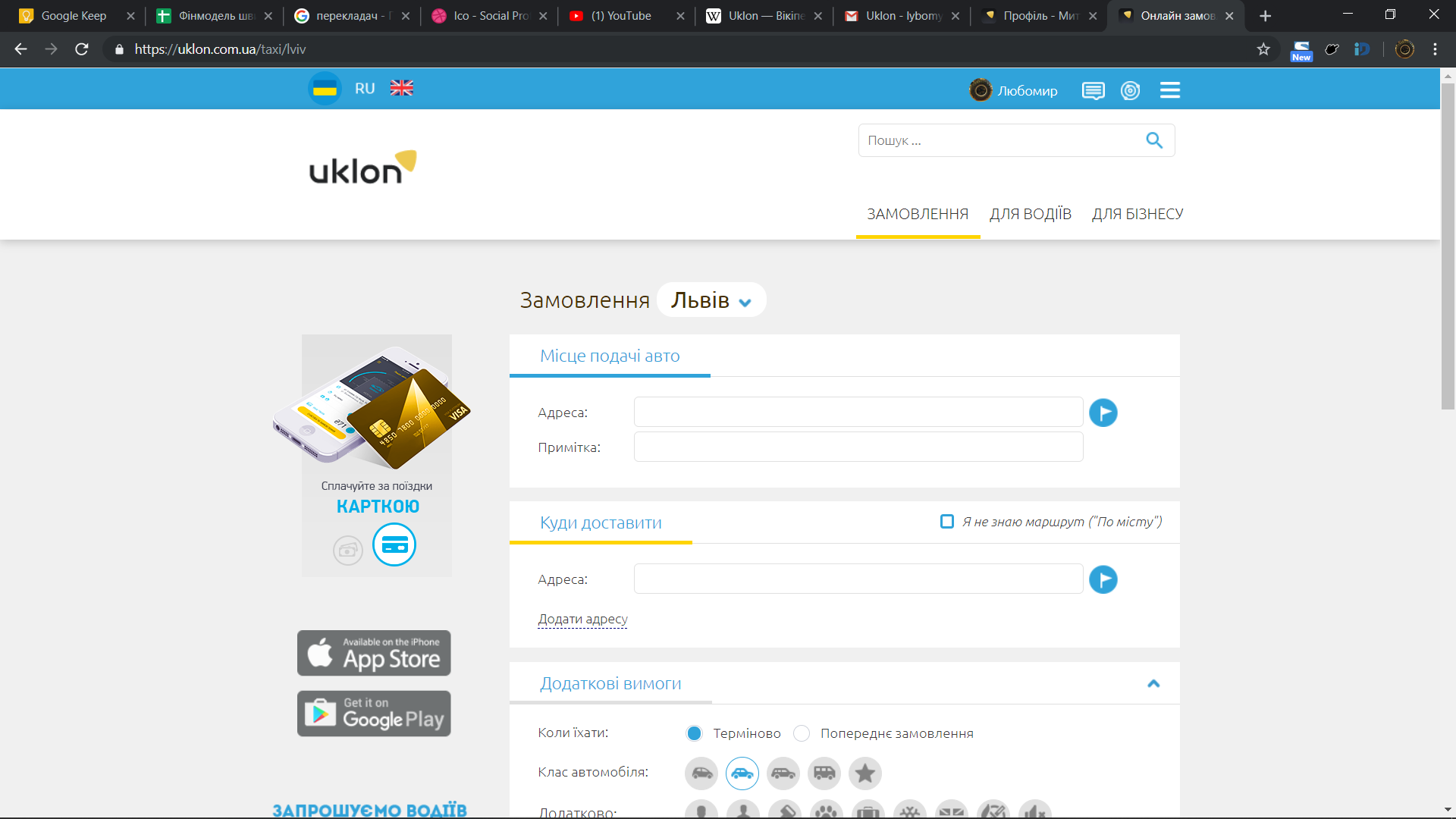
В) Агрегатори з розподілом замовлень змішаного типу. Найменш поширений варіант, який надає користувачу вибір між прийняттям самостійного рішення та автопідбором. Такий варіант найчастіше зустрічається в системах, де кількість чи якість вхідних даних не може гарантувати оптимального результату при автоматичному підборі, тому як альтернатива використовується ручний вибір.

## **1.2. Короткий огляд систем-аналогів**

В даній бакалаврській кваліфікаційній роботі обрано напрямок на вузьконаправлені сервіс-агрегатори з автоматизованим розподілом замовлень. Нижче розглянуто кілька вищезгаданих систем для розуміння їх основних переваг та недоліків.

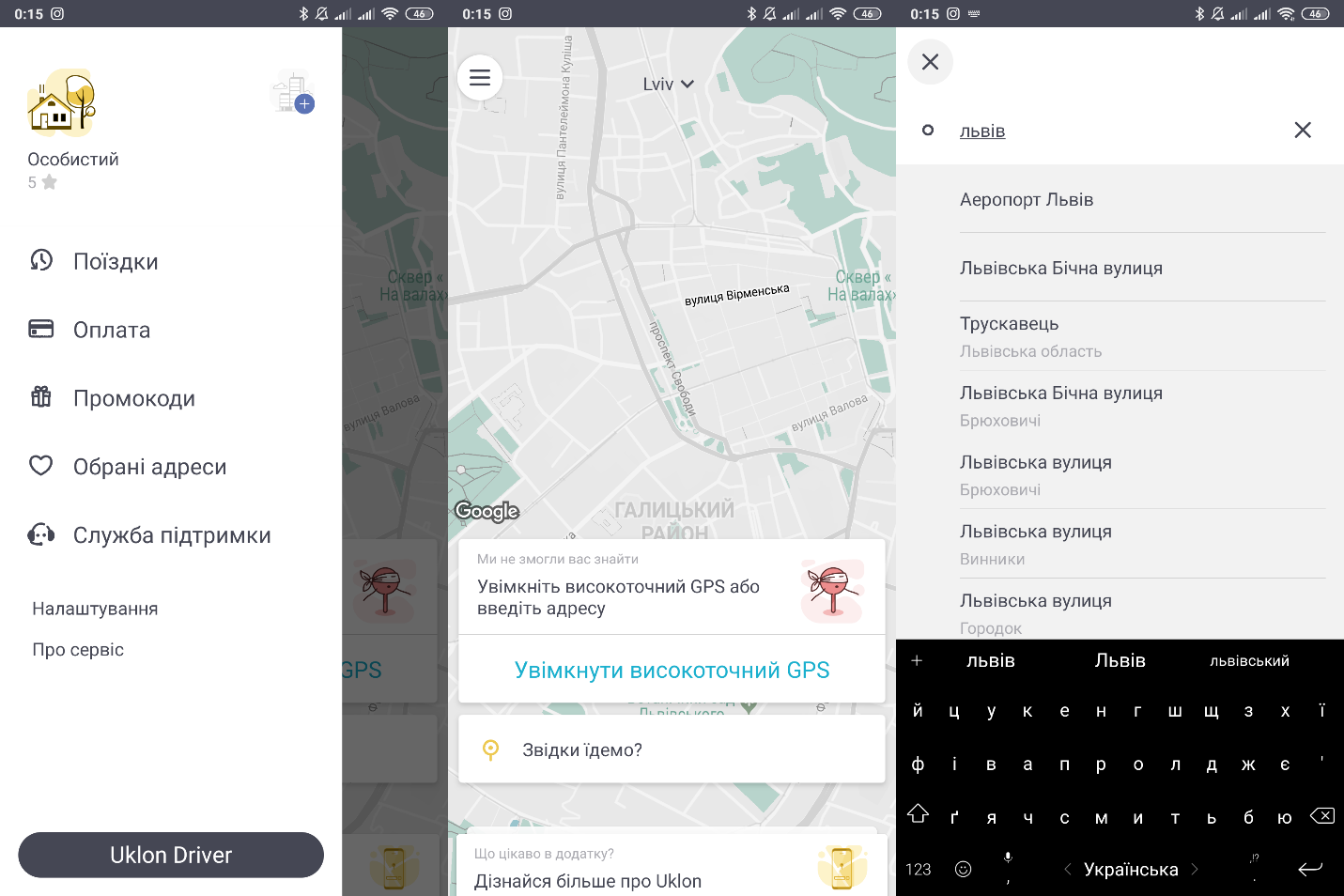
**Uklon**

Сервіс із замовлення таксі Uklon дозволяє користувачеві замовити машину з водієм та відстежити її переміщення до зазначеного місця, оплата послуг водія проводиться готівкою або банківською карткою. Додаток доступний для операційних систем iOS та Android, а також існує веб-версія (рис. 1.1) для людей, які не користуються смартфонами.



*Рис.1.1 – Інтерфейс веб-версії Uklon*

Сервіс дозволяє замовити таксі без необхідності дзвонити до диспетчерських служб: через застосунок зі смартфону (рис. 1.2) або на веб-сайті. Uklon не є службою таксі — це агрегатор замовлень, що розподіляє виклики між водіями-таксистами та службами. Послуги перевезення пасажирів надають, зазвичай, диспетчерські служби таксі. Віднедавна сервіс розпочав набір і власних водіїв. Розподіл замовлень відбувається в напівавтоматичному режимі – водій може обрати клієнта самостійно, або скористатись функцією автопошуку за найоптимальнішими параметрами.



*Рис.1.2 – Інтерфейс додатку Uklon для Android*

На даний момент це один із найбільш часто використовуваних сервісів в українському сегменті послуг таксі, поруч з Uber та Bolt.

Рівень якості користувацького інтерфейсу мобільних додатків та сайту Uklon можна оцінити як задовільний, оскільки деколи виникають проблеми з навігацією, а дизайнерські рішення не найкращим чином мотивують користувача взаємодіяти з інтерфейсом.

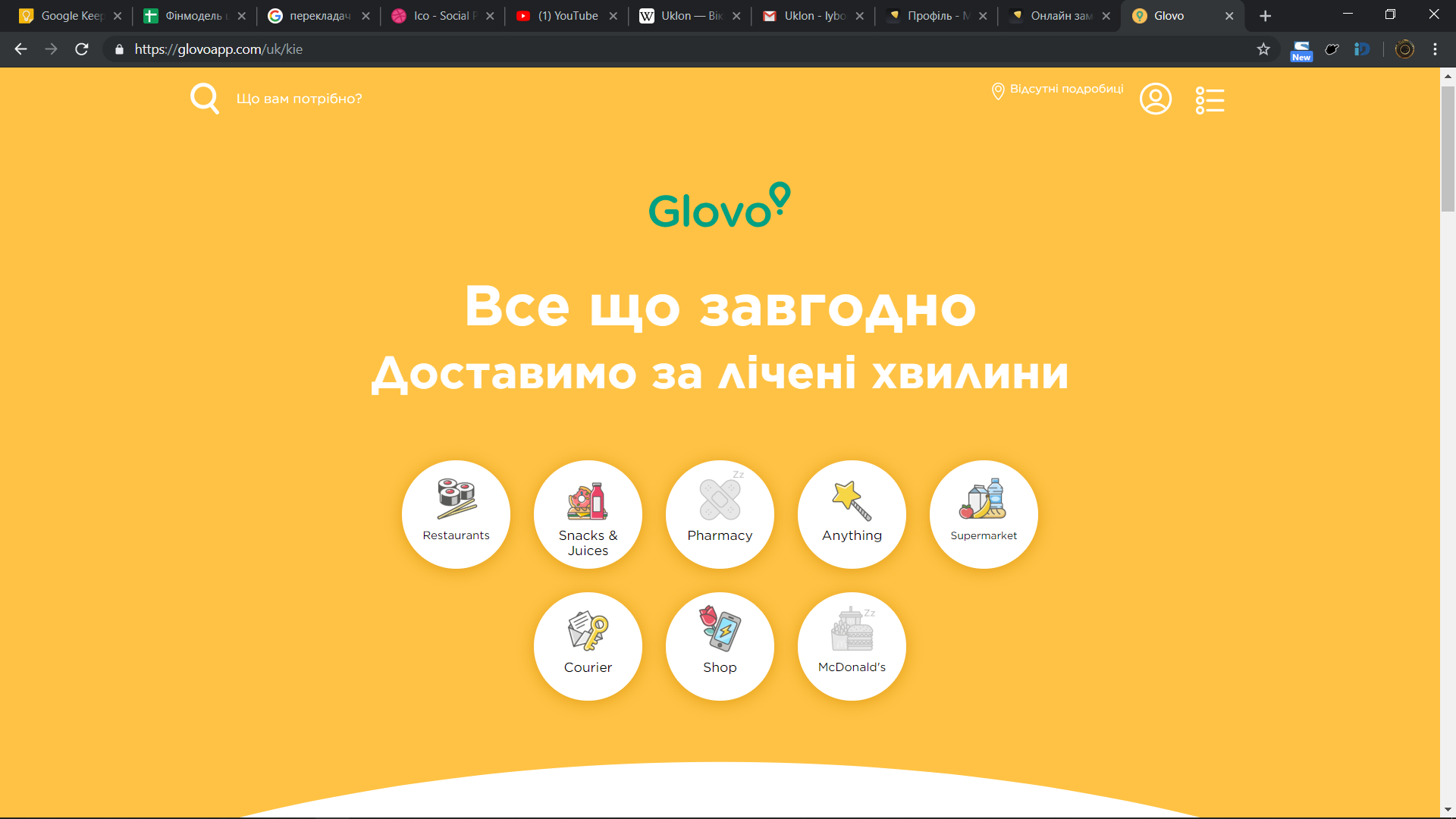
Бізнес-аналітика як така в Uklon відсутня – водію доступна лише стисла статистика заробітку за попередній період часу та історія замовлень. З іншого боку, в цій галузі детальна аналітика не є настільки потрібною, як в деяких інших галузях, тому, вважати це недоліком невиправдано.

Монетизація в Uklon реалізована шляхом списування комісії у вигляді відсотку від оплати за кожне перевезення, що є стандартною практикою в даній галузі. Таке рішення не вимагає жодних додаткових витрат для клієнта, а для водія ці витрати є не дуже суттєвими, отже монетизація реалізована достатньо збалансовано.

**Glovo**

Glovo— іспанський стартап, сервіс для доставки речей з автоматизованим розподілом замовлень. Станом на 2019 рік компанія працює у понад 170 містах в 24 різних країнах. З сервісом співпрацює більш ніж 25 000 кур'єрів.

Через мобільний додаток або веб-сайт (рис. 1.3) клієнт може замовити будь-який товар невеликого розміру (якщо його габарити не перевищують 40x40x30см) та вагою до 9 кг. Зазвичай це їжа, ліки, та документи. В середньому близько 85% всіх замовлень через сервіс припадають на покупку та доставку їжі.

*Рис.1.3 – Інтерфейс веб-версії Glovo*

Після оформлення замовлення користувач має можливість відстежувати переміщення кур'єра по маршруту в режимі реального часу. Основний транспорт кур’єрів – велосипеди, скутери та мотоцикли. Працюють також і піші кур’єри.

Сервіс доступний на iOS, Android (рис. 1.4) та у вигляді веб-сайту.



*Рис.1.4 – Інтерфейс додатку Glovo для Andoid*

В Україні сервіс запустився відносно недавно, тому поки що тільки набуває популярності.

Веб-інтерфейс виглядає приємно, але має значні проблеми із зручністю користування та навігацією – виконання деяких дій є надзвичайно неочевидним і вимагає від користувача значно більше часу ніж це є допустимим для систем із такою великою популярністю у світі. Інтерфейс додатку реалізовано краще, але тим не менше, в ньому все одно повторюються деякі проблеми, які наявні у веб-версії.

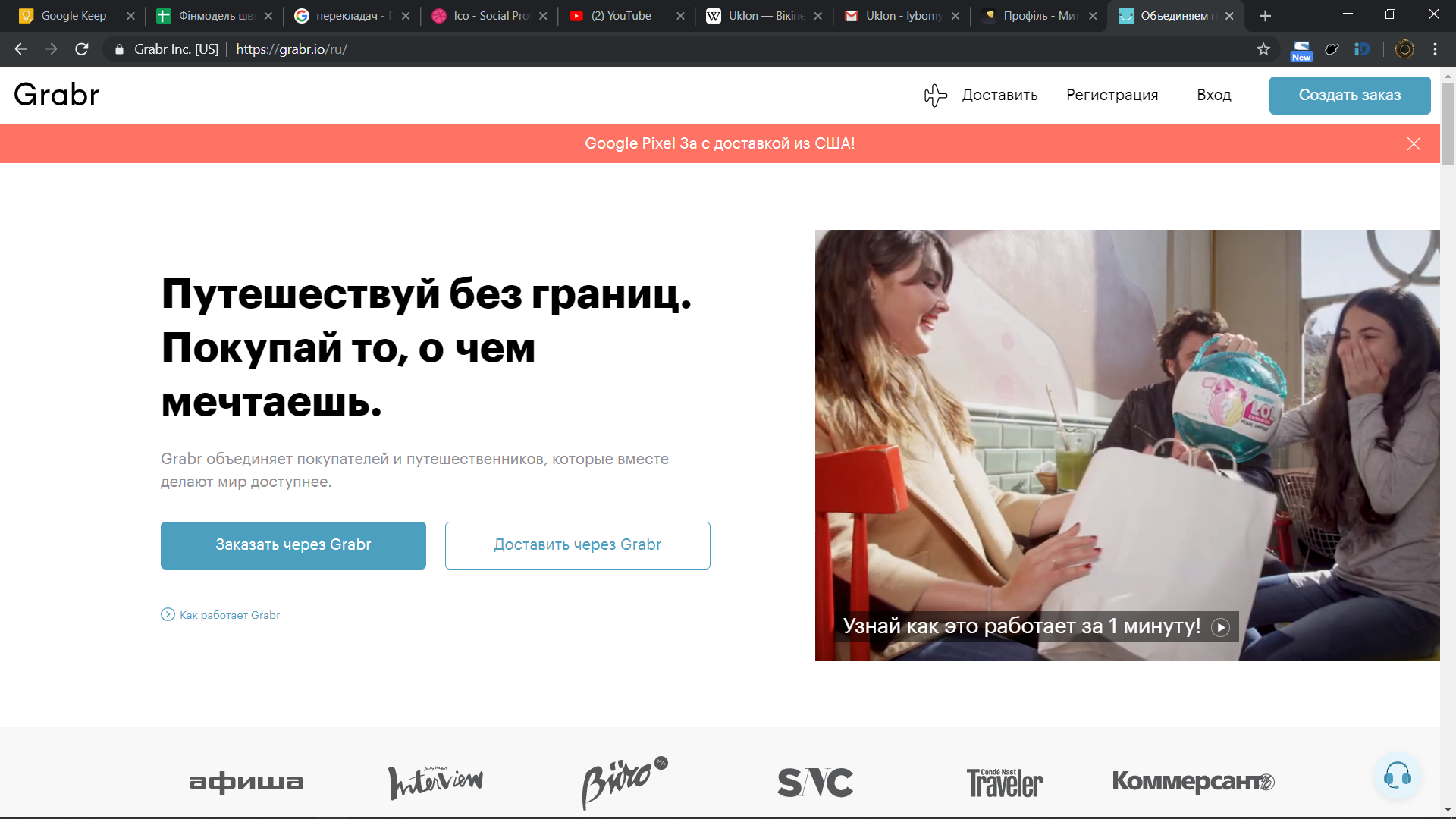
Монетизація відбувається завдяки тому що користувач сплачує певну суму за доставку (вона, зазвичай, залежить від об’єкту та локації доставки) – в цю суму включена оплата послуг кур’єра та комісійний відсоток агрегатора. Як і у випадку з Uklon, принцип монетизації класичний для цієї галузі і не створює суттєвих незручностей ні користувачеві, ні кур’єру, і при цьому завдяки масовості приносить великі прибутки агрегатору. Узагальнюючи, все це означає що система монетизації в даному випадку добре збалансована та може слугувати взірцем.

В системі доступна стисла аналітика для кур’єрів, щоб вони могли відслідковувати ефективність своєї роботи та прибутки за обраний період часу. Такий необширний функціонал є цілком виправданим, оскільки кур’єри є лише працівниками сервісу, а не представниками агрегованого бізнесу. До того ж, аналітику для бізнесу складно реалізувати, оскільки вся взаємодія сервісу і бізнесу відбувається оффлайн, тому її важко відслідковувати.

**Grabr**

Grabr – це сервіс для доставки речей туристами з-за кордону. Розподіл замовлень здійснюється повністю вручну – людині, якій потрібна доставка необхідно розмістити завдання про доставку певного товару з певної країни, а туристи, які подорожують до цієї країни можуть пропонувати обсяг оплати за доставку та брати замовлення. Grabr здійснює посередництво між цими людьми, і бере за це комісію в обсязі 10% від суми доставки – ці кошти доведеться сплатити людині, що замовляє доставку, на додачу до власне вартості доставки.

Інтерфейс веб-сайту (рис 1.5) виконаний в простій світлій стилістиці і є інтуїтивно зролумілим та легким для навігації. Серед трьох представлених аналогів Grabr є найбільш зручним та дружнім до користувача. Інтерфейс мобільного додатку повністю повторює стилістику та простоту в користуванні які реалізовані на веб-сайті. Незважаючи на всі переваги, інтерфейс Grabr не є яскравим та не привертає увагу з першого погляду, як інтерфейс інших аналогічних систем.

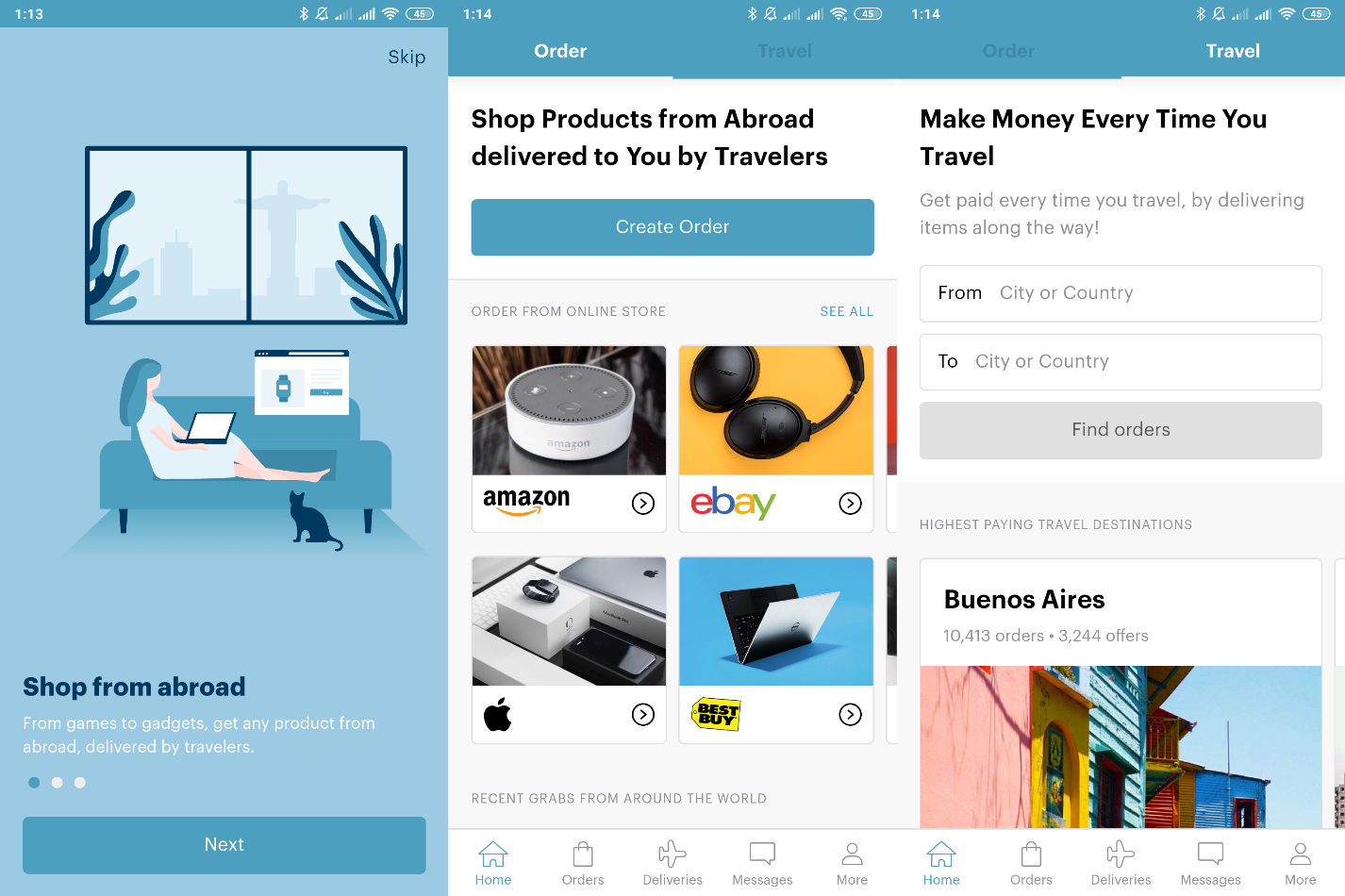


*Рис.1.5 – Інтерфейс веб-версії Grabr*

З іншого боку, інтерфейсом, що реалізований в стриманій та спокійній кольоровій гаммі значно приємніше користуватися, тому не можна назвати таке рішення суттєвою проблемою. Ще одна перевага – «картковий» дизайн, який дуже добре структурує контент та робить простою адаптацію сайту під мобільні пристрої.

Монетизується сервіс повністю за рахунок замовників – окрім самої суми доставки потрібно додатково сплатити 10% від її вартості. Такий принцип є далеко не найкращим – часто ця додаткова сума є причиною відмовитися від користування сервісом. Оптимальнішим варіантом було б включати комісію в суму доставки, при тому щоб її сплачував частково замовник, а частково кур’єр-турист. Втрати агрегатора від переходу на такий принцип монетизації були б несуттєвими, або ж повністю компенсувались би збільшенням кількості замовлень завдяки збільшенням зручності та вигоди для користувачів сервісу. Але на жаль, на даний момент монетизація реалізована достатньо незбалансовано і вимагає від замовника значних додаткових витрат, що тільки відлякує нових користувачів.

Grabr доступний у вигляді сайту та як додаток для iOS і Android (рис. 1.6).



*Рис.1.6 – Інтерфейс додатку Glovo для Andoid*

Бізнес-аналітики не передбачено ні для туристів, ні для замовників. Основна причина – її непотрібність в даній системі. Присутня лише історія замовлень та невеликий особистий кабінет.

**Висновки:**

Надано класифікацію агрегаторних систем за різними чинниками. Проведено аналітичний огляд кількох популярних агрегаторів з різних категорій діяльності. Визначено переваги та недоліки інтерфейсу, монетизації та бізнес-аналітики для кожного з прикладів.

# **2. АНАЛІЗ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ПРОЕКТУ**

Реалізація завдання моєї дипломної бакалаврської кваліфікаційної роботи є комплексним та складним процесом і вимагає розуміння та практичних навичок роботи з багатьма різними технологіями. Нижче описано більшість технологій, використаних при виконанні завдання.

## **2.1 Основні особливості розробки інтерфейсів**

Графічний інтерфейс користувача є важливою частиною будь-якої програми або системи. Від інтерфейсу залежить швидкість освоєння системи користувачем і якість виконання роботи. Графічний інтерфейс повинен звести до мінімуму можливість допущення помилок користувачем. Тому, до розробки інтерфейсу потрібно підходити серйозно та виділяти більше часу на його проектування.

Вимоги до ідеального інтерфейсу, зазвичай, відрізняються у різних категорій людей. Але існують загальні принципи, яких завжди потрібно дотримуватися. Основний принцип – інтерфейс завжди повинен бути інтуїтивно зрозумілим. Він повинен бути самодокументованим, очевидним, та обов’язково не повинен бути нав'язливим. Так, наприклад, не варто набридати користувачеві в прагненні допомогти, виводячи на екран занадто велику кількість підказок. З іншого боку, повинен забезпечуватися належний контроль даних, які вводить користувач.

### **2.1.1 Інтуїтивно зрозумілий інтерфейс**

Інтерфейс можна назвати інтуїтивно зрозумілим, коли у користувача не виникає жодних проблем із роботою з ним без попередньої підготовки та вивчення. Користувач відразу розуміє, на які кнопки натискати щоб виконати необхідну дію. У всіх користувачів свій унікальний досвід роботи з програмами та їх інтерфейсами. Саме тому, при проектуванні інтерфейсів варто використовувати загальновідомі принципи та практики, і далеко не завжди нові унікальні рішення є виправданими.

Для очевидності інтерфейсу важливо щоб користувачеві були доступні всі елементи управління, які необхідні для виконання завдання на поточному етапі. На окремому етапі користувачеві, швидше за все, не знадобиться весь функціонал системи системи, тому зайві елементи можна приховати. Водночас із цим, елементів потрібних для виконання однієї конкретної задачі не повинно бути занадто багато – варто витримувати максимальну простоту і слідкувати за балансом між зовнішнім виглядом та зручністю використання.

### **2.1.2 Самодокументований інтерфейс**

Самодокументований інтерфейс означає, що система надає користувачеві певну допомогу в освоєнні у вигляді розгорнутих пояснень до елементів управління. Ці пояснення можуть як просто виводитися безпосередньо поруч з елементами управління, так і бути реалізованими у вигляді спливаючих підказок. Це допоможе користувачеві допустити меншу кількість помилок та швидше навчитись користуватись складною системою.

Важливу роль відіграють повідомлення користувачу про успішне виконання операції або помилку при її виконанні. При цьому, коли виникає помилка, слід пояснювати користувачу чому вона виникла та яким чином можна уникнути цього наступного разу. Варіантом уникнення сповіщень про проблеми є превентивне вирішення помилок, і іноді їх використання може суттєво покращити досвід взаємодії з інтерфейсом.

### **2.1.3 Ненав’язливий інтерфейс**

Інтерфейс не повинен бути надмірно нав'язливий в своєму прагненні допомогти. Так, наприклад, буде вкрай незручно, коли при виконанні завдання система постійно запитує підтвердження кожної операції – робота з таким інтерфейсом забиратиме занадто багато часу. Тому, повинен бути передбачений відповідний спосіб видачі підтверджень, що не змушує користувача робити зайвих дій. Такий самий принцип варто застосовувати і при виведення підказок – вони повинні з’являтися рівно до того моменту поки користувач не почне впевнено користуватися інтерфейсом. Як альтернатива – можна дати користувачу вибір між тим щоб показувати підказки чи приховувати їх, залежно від його бажання та оцінки власних можливостей.

### **2.1.4 Перевірка даних**

При роботі з системою користувачеві допомагає ненав'язлива перевірка даних. Перевірка введених даних повинна бути передбачена на кожному етапі як превентивне вирішення виникнення проблем та помилок.

На першому етапі відбувається робота з певними елементами управління. Тут повинна бути реалізована перевірка на правильність заповнення форм. Наприклад, перевірка умови, коли потрібно вводити тільки натуральне число, або умови, коли значення поля повинно бути унікальним, і не збігатися зі значенням цього ж поля, але іншого елемента.

На наступному етапі повинна відбуватися робота на рівні системи. Наприклад, вхідні дані повинні перевірятися на наявність шкідливих конструкцій для системи для їх подальшого безпечного запису в базу даних. Після цього повинні бути сформовані повідомлення про помилки (в разі їх наявності) до потрібних форм і видані користувачу системи у вигляді спливаючих вікон чи подібних елементів які привертають увагу.

### **2.1.5 Мінімізація помилок користувача**

Інтерфейс повинен звести до мінімуму помилки користувача. Для цього потрібно використовувати підказки і перевірки. Все це допоможе мінімізувати кількість помилок.

## **2.2 Проектування інтерфейсу**

Проектування інтерфейсу можна умовно поділити на такі кроки:

* Скласти портрет користувача і список його цілей. Цілі можна розбити на завдання, які користувачеві треба буде розв'язати, і скласти список завдань для кожної мети.
* Написати користувацькі сценарії відповідно до списків завдань. Сценарії повинні мати вигляд «користувач зробив - система відповіла».
* Скласти список сторінок інтерфейсу відповідно зі сценаріями.
* Скласти структуру навігації, тобто послідовність виконання сценаріїв. Необхідно поставити акцент на важливі сценарії.
* Описати всі сторінки інтерфейсу.
* Відповідно до інформації сформовано раніше розробити дизайн всіх сторінок
* Оцінити дизайн та виправити можливі помилки

## **2.3 Технології розробки інтерфейсів**

Технології розробки інтерфейсів відрізняються в залежності від конкретної платформи. Далі розглянуто технології, які використовуються при розробці веб-інтерфейсів та інтерфейсів для додатків на базі операційної системи Android.

Для веб-інтерфейсів класичною зв’язкою технологій є HTML, CSS та JavaScript. Окремо кожна з них не має великого сенсу, але при їх спільному використанні можна розробляти комплексні інтерфейси.

**HyperText Markup Language** (**HTML**) – мова розмітки гіпертекстових документівв - класична [мова розмітки](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D1%80%D0%BE%D0%B7%D0%BC%D1%96%D1%82%D0%BA%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85) для верстки [веб-сторінок](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D0%BE%D1%80%D1%96%D0%BD%D0%BA%D0%B0) і [веб-додатків](https://uk.wikipedia.org/w/index.php?title=%D0%92%D0%B5%D0%B1-%D0%B4%D0%BE%D0%B4%D0%B0%D1%82%D0%BA%D1%96%D0%B2&action=edit&redlink=1). HTML описує структуру веб-сторінки [семантично](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BC%D0%B0%D0%BD%D1%82%D0%B8%D1%87%D0%BD%D0%B0_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D0%B0) і представляє собою набір тегів призначених для різного контенту, від порядку яких залежить власне візуальний порядок елементів на сторінці.

[Базові елементи HTML](https://uk.wikipedia.org/wiki/%D0%95%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%B8_HTML) – теги – є будівельними блоками сторінок HTML. HTML надає засоби для створення [структурованих документів](https://uk.wikipedia.org/w/index.php?title=%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%85_%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D1%96%D0%B2&action=edit&redlink=1), позначаючи структурну [семантику](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BC%D0%B0%D0%BD%D1%82%D0%B8%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) тексту, наприклад заголовки, абзаци, списки, [посилання](https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D0%B8%D0%BB%D0%B0%D0%BD%D0%BD%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), цитати та інші елементи. Теги HTML пишуться з використанням кутових дужок. Більшість тегів мають відкриваючу та закриваючу частини, наприклад <div>content</div>, але існує невелика кількість тегів які не потрібно закривати. Браузери не показують теги HTML, але використовують їх для інтерпретації вмісту сторінки.

HTML має можливість інтеграції програм, написаних на скриптових мовах, наприклад [JavaScript](https://uk.wikipedia.org/wiki/JavaScript), що впливає на ситуативну поведінку та контент веб-сторінок. Додавання CSS визначає вигляд і компонування контенту.

**Cascading Style Sheets (CSS)** — каскадні таблиці стилів — спеціальна мова, що використовується для опису зовнішнього вигляду сторінок, написаних мовами розмітки даних.

CSS використовується розробниками веб-сторінок, щоб задати кольори, шрифти, стилі та інші аспекти вигляду сторінки. Одна з найголовніших переваг — можливість розділити зміст сторінки (або контент, наповнення, зазвичай HTML, XML або подібна мова розмітки) від вигляду документу (що описується виключно в CSS).

Таке розділення покращує сприйняття та доступність контенту, забезпечує більшу гнучкість та контроль за відображенням контенту на різних пристроях та в різних браузерах, робить контент набагато більш структурованим та простим, прибрає повтори тощо. До того ж, один HTML або XML документ може бути відображений по-різному в залежності від використовуваного CSS – таким чином можна реалізувати вибір між різними «темами» для користувачів.

**JavaScript (JS)** — динамічна, об'єктно-орієнтована прототипна мова програмування. Найчастіше використання - створення cкриптів веб-сторінок, що надає можливість на стороні клієнта (пристрої кінцевого користувача) інтерактивно взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру, зовнішній вигляд та контент сторінки.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

JavaScript та різноманітні фреймворки на його основі використовується в багатьох цілях:

* написання скриптів веб-сторінок для надання їм інтерактивності;
* створення односторінкових веб-додатків (React, AngularJS, Vue.js);
* програмування на стороні сервера (Node.js);
* створення стаціонарних застосунків (Electron, NW.js);
* мобільних додатків (React Native, Cordova);
* сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite чи Apache JMeter);
* всередині PDF-документів тощо.

Аналогом HTML для мобільних додатків на базі операційної системи Android є **Extensible Markup Language (XML) –** розширювана мова розмітки. Базові принципи (теги, ієрархічність тощо) аналогічні до принципів HTML, але синтаксис частково відрізняється. Як і HTML, XML-документ повинен мати лише один кореневий елемент, а решта тегів мусять бути ієрархічними піделементами кореневого елементу.

Певні веб-браузери здатні безпосередньо відображати XML-документи. Це може досягатись завдяки використанню таблиць стилів. Вказані у таблиці стилів операції можуть призводити до перетворення XML-документа в інший, відмінний від XML формат.

Як і для HTML, специфікації та стандарти для XML видаються комітетом W3C (World Wide Web Consortium).

Для спрощення реалізації інтерфейсу в своїй бакалаврській кваліфікаційній роботі використано клієнтський фреймворк **Bootstrap** – бібліотеку готових CSS-класів та JS-скриптів, яка значно пришвидшує розробку веб-сторінок та суттєво спрощує адаптацію під мобільні пристрої.

Основний принцип розробки веб-сторінок на цьому фреймворку – розподіл контенту по контейнерах, рядках та колонках і менеджмент параметрів цих класів за допомогою так званих брейкпойнтів – «точок зламу».

## **2.4 Технології розробки бізнес-аналітики**

Будь-яка аналітична система, зазвичай, працює за одним принципом – вона збирає детальну статистику, обробляє зібрані дані та відображає в зручному вигляді через користувацький інтерфейс.

Статистика – це завжди впорядковані дані. А впорядкований набір даних завжди реалізовується через бази даних.

**MySQL** – безкоштовна система керування реляційними базами даних з відкритим кодом. Вона дозволяє як зберігати дані в базах даних, так і робити досить комплексні запити, які важливі для бізнес-аналітики. Система може забезпечувати компактний багатопотоковий сервер баз даних. Характеризується високою швидкістю, стійкістю і простотою використання.

MySQL вважається вдалим рішенням для малих і середніх застосувань. Вихідні коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатопоточності, що підвищує продуктивність системи в цілому.

Можливості сервера MySQL:

• простота у встановленні та використанні;

• підтримується необмежена кількість користувачів, що одночасно працюють із БД;

• кількість рядків у таблицях може досягати 50 мільйонів;

• висока швидкість виконання команд;

• наявність простої та ефективної системи безпеки.

База даних MySQL - реляційна база даних. У реляційній базі даних дані зберігаються не всі разом, а в окремих таблицях, завдяки чому досягається виграш у швидкості та гнучкості. Таблиці зв'язуються між собою за допомогою зв’язків, завдяки чому забезпечується можливість об'єднувати при виконанні запиту дані з декількох таблиць. SQL як частину системи MySQL можна охарактеризувати як мову структурованих запитів та найбільш поширену стандартний мову, що використовується для доступу до баз даних.

Система управління базами даних MySQL є системою клієнт-сервер, яка містить багатопоточний SQL-сервер, що забезпечує підтримку різних обчислювальних машин, а також кілька різних клієнтських програм і бібліотек, засоби адміністрування і широкий спектр програмних інтерфейсів (API).

Сервер MySQL постійно працює на хостингу. Клієнтські програми (наприклад, скрипти PHP) надсилають серверу MySQL SQL-запити через механізм сокетів (тобто за допомогою мережевих засобів), сервер їх обробляє і запам'ятовує результат. Тобто, скрипт (клієнт) вказує, яку інформацію він хоче отримати від сервера баз даних. Потім сервер баз даних надсилає відповідь (результат) клієнтові (скрипту).

Структура MySQL трьохрівнева, як і у більшості реляційних баз даних:

а) бази даних;

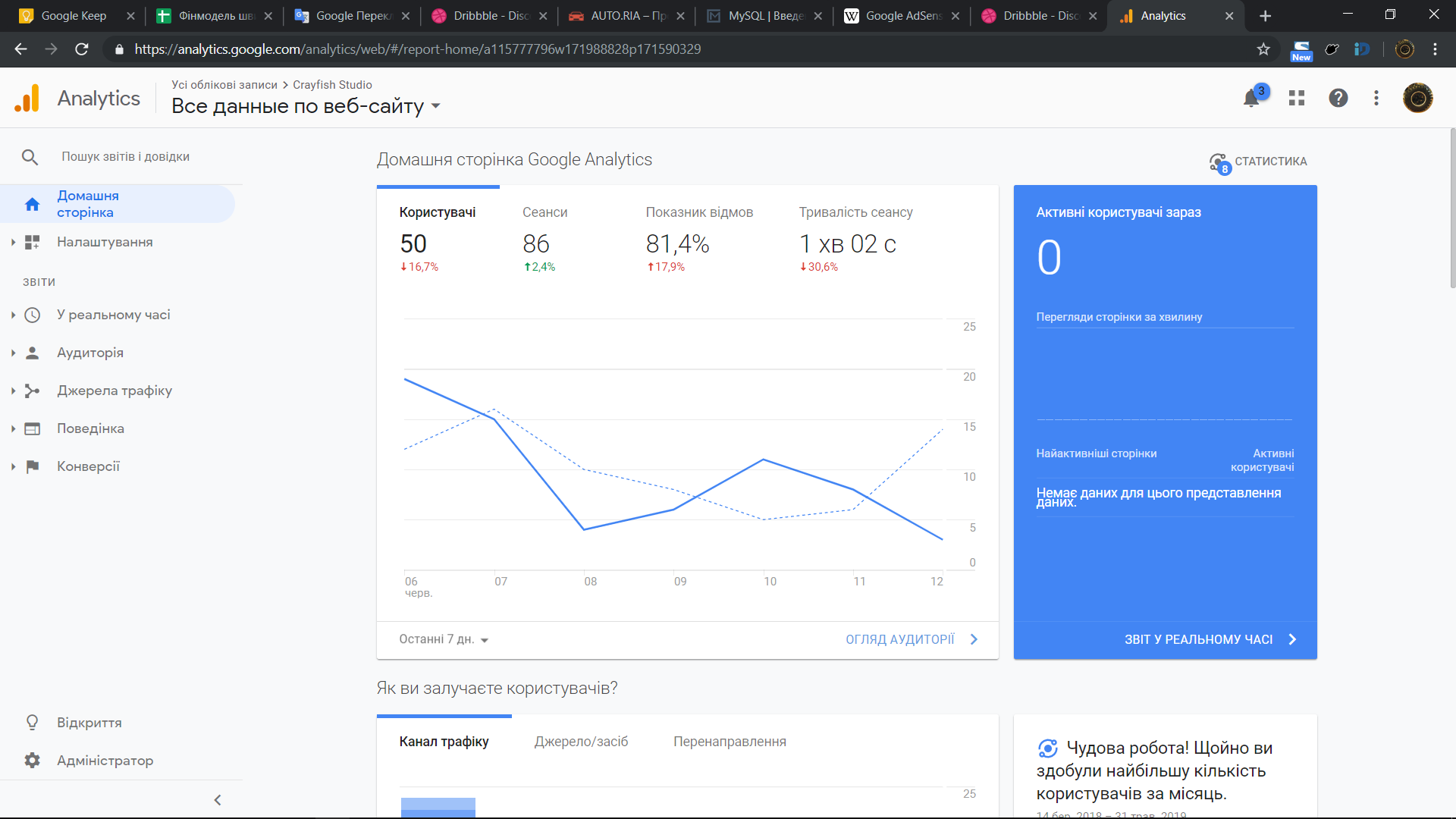
б) таблиці;

в) записи.

MySQL є додатком з відкритим кодом. Кожен користувач може вивчити вихідний код і змінити його у відповідності зі своїми потребами.

Основні переваги пакету MySQL:

* Багатопотоковість. Підтримка декількох одночасних запитів.
* Оптимізація зв'язків з приєднанням багатьох даних за один прохід.
* Записи фіксованої і змінної довжини.
* ОDBC драйвер в комплекті з вихідним текстом.
* Гнучка система привілеїв і паролів.
* До 16 ключів в таблиці. Кожен ключ може мати до 15 полів.
* Заснована на потоках, швидка система пам'яті.
* Всі операції роботи з рядками не звертають уваги на регістр символів в оброблюваних рядках



*Рис.2.1 – Веб-інтерфейс Google Analytics*

Окрім варіанту самостійної розробки, існують готові рішення для аналітики відвідуваності та конверсій. Беззаперечний лідер в цьому – **Google Analytics** (рис. 2.1).

Google Analytics вміє збирати статистику відвідувань сайту або використань додатку, визначати джерела трафіку, кількість активних користувачів, кількість поточних користувачів, розподіл користувачів по пристроях, демографії та багатьох інших показниках:

* вік;
* стать;
* зацікавлення;
* геодані;
* технології (пристрій, операційна система, браузер тощо);
* поведінка.

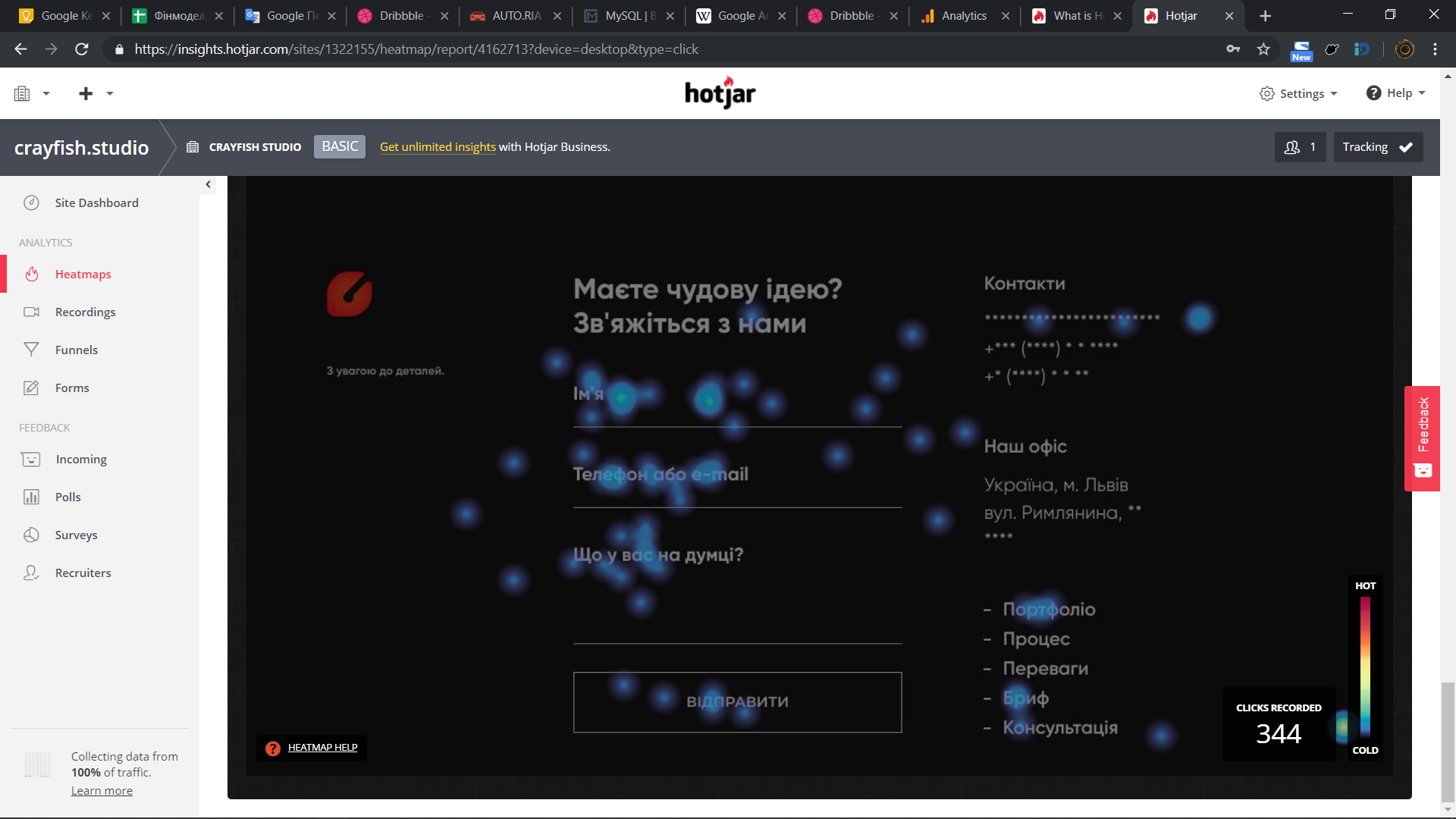
Ці дані дозволяють отримати краще розуміння поточної аудиторії сайту чи додатку, і відповідно побачити та виправити всі вузькі місця та недоліки.

Важливою особливістю сервісу є можливість відслідковування цілей та конверсій. Наприклад, розробник може визначити за мету 1000 відвідувань сайту в місяць та 2% конверсії, тобто 20 відправок контактної форми – тоді Google Analytics буде постійно відслідковувати ці показники, повідомляти про них розробнику та надавати поради щодо оптимізації для досягнення заданого результату.

Якщо потрібна не загальна статистика відвідувань, а детальна аналітика поведінки користувача на сайті, то доцільно використовувати **Hotjar** – це відносно новий, потужний інструмент, який за допомогою теплових карт та запису екрану під час перегляду дозволяє розробнику безпосередньо побачити поведінку користувача на веб-сторінці. Об'єднуючи інструменти аналізу та зворотного зв'язку, Hotjar надає детальну картину того, як покращити досвід користувачів і показники ефективності сайту.

На відміну від Google Analytics, Hotjar не надає майже ніякої статистики відвідувань, зате він має два дуже потужних інструменти, яких немає у конкурента – теплові карти та відеозаписи сесій відвідувань сайту користувачами.

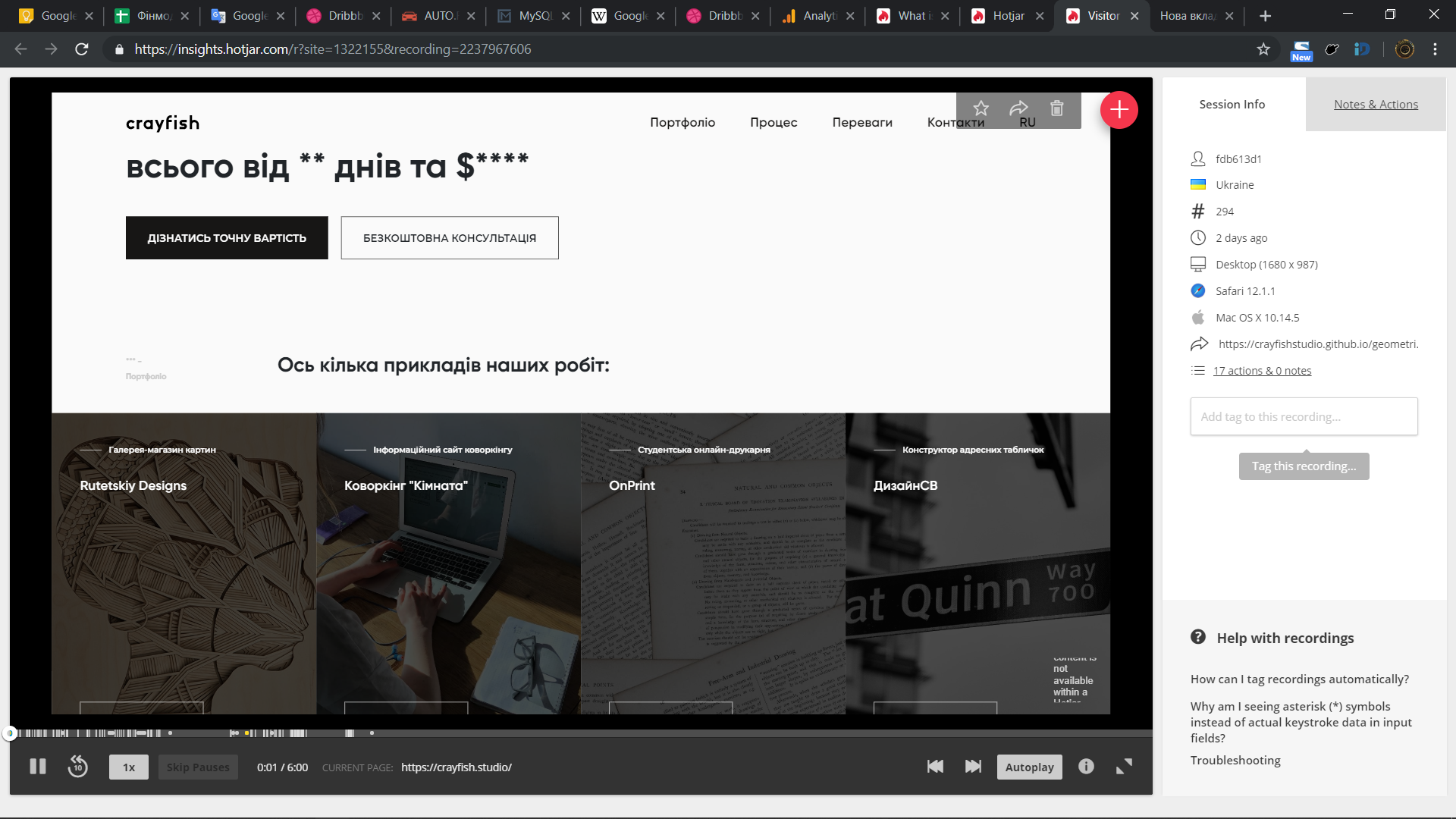
Теплова карта (рис. 2.2) показує кількість кліків по різних ділянках на сторінці, що дозволяє легко оцінити інтуїтивність та зручність інтерфейсу. Генерація теплових карт проводиться окремо для трьох розмірів екранів – ноутбук, планшет та смартфон. Завдяки цьому можна проаналізувати, виправити помилки та оптимізувати користувацький досвід для кожної окремої платформи.



*Рис.2.2 – Приклад теплової карти в системі Hotjar*

Окрім кількості кліків, Hotjar записує теплові карти скролінгу сторінки, завдяки чому можна побачити скільки відсотків користувачів долистали сторінку до певного моменту і на основі цього зробити висновки про те, на якому моменті користувач втрачає зацікавленість або стикається з певною проблемою в користуванні інтерфейсом. Як і теплові карти кліків, скрол-карти записуються для трьох різних платформ окремо. Це виправдане рішення з точки зору розробників системи та зручності користування, оскільки зазвичай досвід користування сайтом з ноутбука значно відрізняється від досвіду того ж користувача того ж сайту при перегляді з мобільного пристрою чи планшета.

Другий потужний інструмент – запис екрану користувача під час відвідування сайту (рис. 2.3).



*Рис.2.3 – Запис сесії відвідування сторінки сайту в системі Hotjar*

Запис відбувається у відеофайл, при тому в інтерфейсі системи на його часовій шкалі позначено активність користувача:

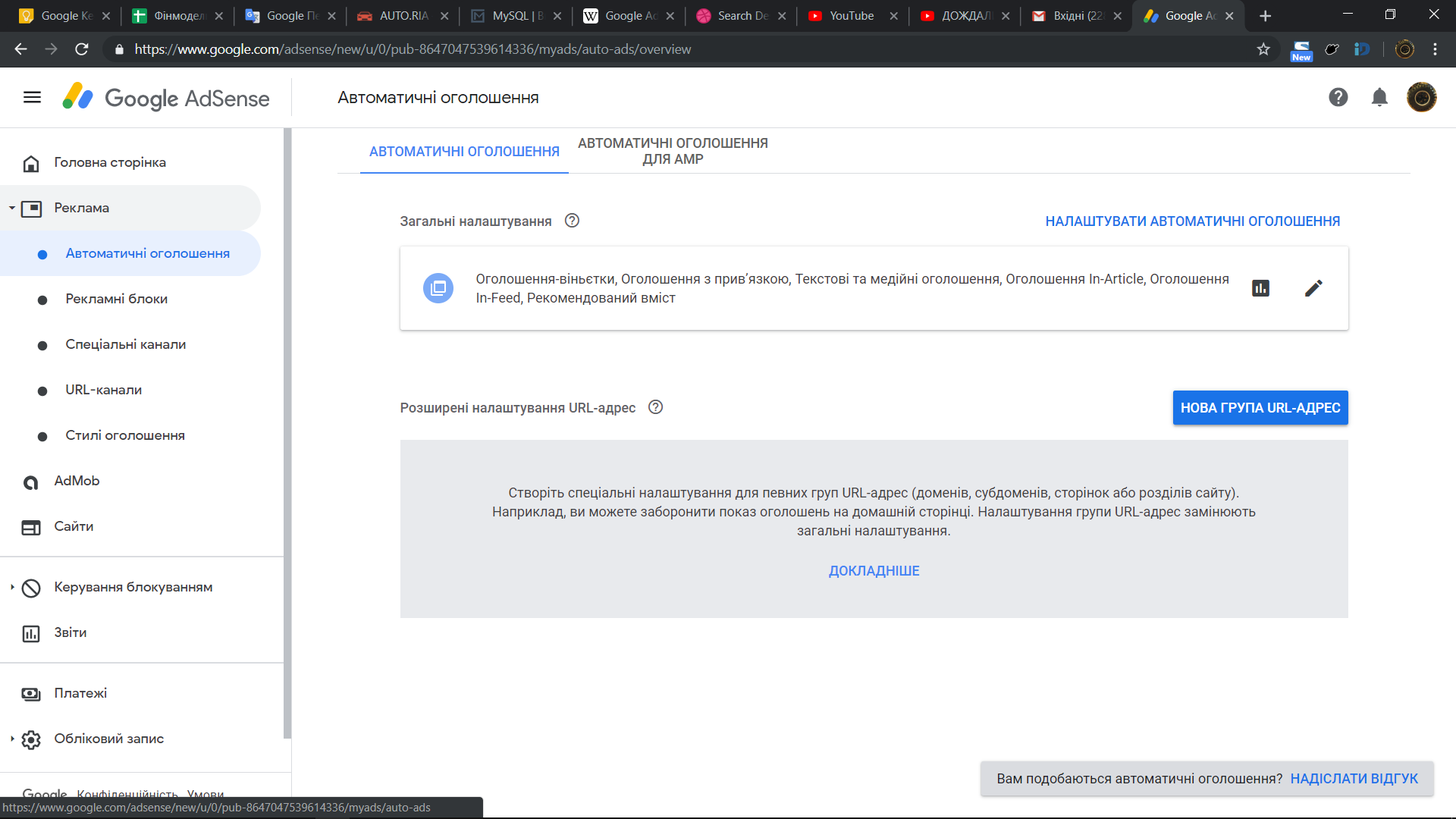
* скролінг сторінки;
* кліки;
* переходи за посиланнями;
* взаємодія з формами.

Такий метод аналітики дозволяє розробнику безпосередньо побачити те, як користувач взаємодіє з його сайтом, і не витрачати час та кошти на дослідження за допомогою фокус-груп чи інших класичних методів.

## **2.5 Технології монетизації**

Монетизувати веб-сайт можна кількома шляхами. В програмному продукті бакалаврської кваліфікаційної роботи монетизацію реалізовано завдяки показу інтегрованих рекламних оголошень та комісійним стягненням з кожної фінансової операції всередині системи. Це означає, що для розробки потрібен сервіс для інтеграції контекстної реклами та платіжна система. Для цих цілей обрано LiqPay для фінансових операцій та Google AdSense для інтеграції контекстної реклами.

**Google AdSense**



*Рис.2.4 – Інтерфейс додавання оголошень в системі Google AdSense*

Google AdSense (рис. 2.4) — сервіс контекстної реклами, що розроблений Google, який у спеціально відведених контейнерах на веб-сторінках автоматично розміщує текстові та графічні оголошення, що тематично підходять за змістом статті, постійно аналізуючи контент самої сторінки, на якій знаходиться оголошення (баннер).

Початковою ідеєю сервісу є показ відвідувачам сайту персоналізовних оголошень, які можуть бути найбільш цікаві кожному конкретному відвідувачу. Іншою особливістю є те що для рекламодавців сервіс дозволяє показувати таргетовану рекламу, тобто націлену виключно на певну категорію користувачів. Такий підхід суттєво здешевлює рекламу та підвищує її ефективність. Окрім цього, система вміє показувати оголошення, які є підібраними на основі відвіданих сайтів.

**LiqPay**

LiqPay - платіжна система, що дозволяє оплачувати рахунки за допомогою віртуального гаманця, який прив'язаний до номера мобільного телефону. Оплата здійснюється за допомогою платіжних карт.

Рахунок LiqPay представляє собою мобільний номер телефону представлений у міжнародному форматі (наприклад, «+380987654321»). Рахунок сервісу прив'язаний до банківської карти Приватбанку, що можна назвати швидше недоліком ніж перевагою, оскільки технічна сторона даного банку останнім часом втрачає свою якість.

* Можливості системи:
* Створення мікроплатежів від 0.02 USD.
* Можливість виведення коштів на VISA-картки та будь-які картки Приватбанку.
* Створення масових платежів.
* Створення платформ API.
* Прийом платежів на сайті.

Оплата банківською карткою з сайту передбачає проведення транзакції через платіжний шлюз банку-еквайра. Користувач оформляє замовлення на сайті продавця, а після вибору кнопки «Оплатити» потрапляє на сторінку шлюзу. Після здійснення платежу покупець автоматично повертається на сайт.

* Оплата картою на сайті продавця має наступні переваги:
* Користувач залишається на сайті до завершення угоди. Це підвищує довіру до продавця і зменшує кількість відмов від завершення покупки.
* Безпеку платежів гарантує банк-еквайєр. Ймовірність крадіжки персональних даних або коштів з рахунку клієнта практично відсутня.
* Вигідні умови обслуговування для клієнтів з великими оборотами.

До недоліків цього способу оплати відноситься складна процедура під’єднання послуги. Для початківців-підприємців і нових інтернет-магазинів витрати можуть виявитися занадто високими.

**Висновки:**

Розглянуто основні технології для виконання поставленого завдання. Оцінено можливості, переваги та недоліки технологій. Перелічено специфічні особливості певних технологій та їх вплив на процес реалізації продукту.

# **3. РЕАЛІЗАЦІЯ ОБРАНИХ МОДУЛІВ СИСТЕМИ**

Реалізація обраних модулів – інтерфейсу, системи монетизації та бізнес-аналітики – це доповнення готового стороннього серверного рішення, що значно підвищує складність виконання завдання через потребу інтеграції у вже існуючу систему. Розроблені модулі і сторонній бекенд, що використовувався разом утворюють платформу (або ж фреймворк) для пришвидшеної розробки товарних та сервіс-агрегаторів з автоматизованим розподілом замовлень. Протягом подальшого пояснення наведено фрагменти структури використаного бекенду.

## **3.1 Реалізація інтерфейсу**

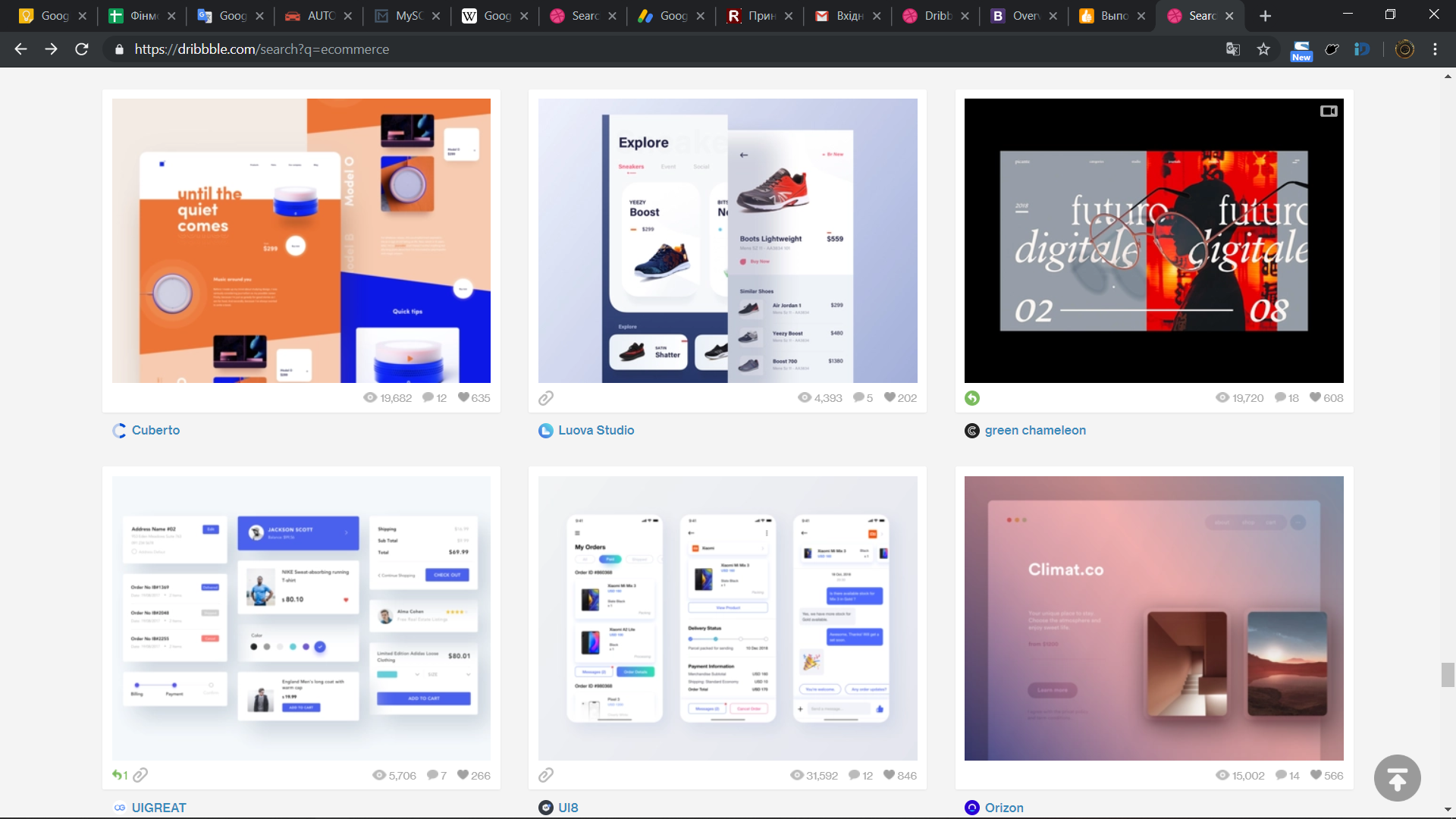
Оскільки припускалося, що сукупність програмних рішень буде використовуватися швидше як універсальний фреймворк а не як одиночний продукт, і часто інтерфейс буде розроблятися окремо, вирішено створити базову бібліотеку елементів інтерфейсу та демо-сторінку для них.

На основі багатьох факторів прийнято рішення розробляти інтерфейс у вигляді веб-сторінок. Основними факторами є доступність для всіх користувачів, простота у реалізації та зручність прив’язки до обраного серверного рішення.

До інтерфейсу поставлено такі вимоги:

* Відповідність функціональним можливостям бекенду.
* Легкий інтуїтивний дизайн.
* Карткова структура.
* Кросбраузерність.
* Адаптивність.
* Мінімізація часу взаємодії користувача із сайтом для виконання поставленої цілі.

Перед початком робіт проведено невелике дослідження інсуючих інтерфейсів в системах із різних галузей, для кращого розуміння сучасних трендів та генерації ідей (рис. 3.1). Дослідження показало, що сучасні інтерфейси у схожих системах стають все лаконічнішими та простішими, тому витримати високу планку якості буде досить складно.



*Рис.3.1 – Дослідження подібних інтерфейсів*

З’ясовано, що незважаючи на покращення загальної ситуації щодо зовнішнього вигляду веб-сайтів, багато з них мають погано збалансоване співвідношення між власне зовнішнім виглядом та зручністю користування. Дизайнери часто роблять акцент не на зручності для користувача, а на ефекті першого враження, через що і страждає інтуїтивність та зрозумілість. В рішенні реалізації інтерфейсу для бакалаврської кваліфікаційної роботи витримано кращий баланс між цими характеристиками ніж у більшості переглянутих при дослідженні прикладів

На першому етапі реалізації інтерфейсу (рис. 3.2) написано користувацькі сценарії виду «користувач зробив – система відповіла» та на основі цього складено список всіх необхідних сторінок. Наступним етапом є прототипування всіх сторінок для попереднього тестування та аналізу. Після цього, на основі отриманих даних, в макети внесено коригуючі правки.



*Рис.3.2 – Алгоритм реалізації інтерфейсу*

В результаті з макетів сторінок взято часто використовувані елементи інтерфейсу і сформовано в так званий UI-kit – базову бібліотеку елементів.

Наступним етапом став дизайн та верстка демо-сторінок. Для спрощення розробки та часткової автоматизації адаптації під мобільні пристрої прийнято рішення використовувати HTML/CSS-фреймворк Bootstrap. Його інтеграція означає додавання в ресурси веб-сайту готових CSS-стилів та JavaScript-сценаріїв.

Основною перевагою Bootstrap є система розподілу контенту по логічних контейнерах (рис. 3.3), яку її розробники назвали «Bootstrap Grid» (сітка Bootstrap). Система розроблялася для «mobile-first» (спершу мобільні пристрої) верстки, але так само добре працює і у зворотньому порядку.

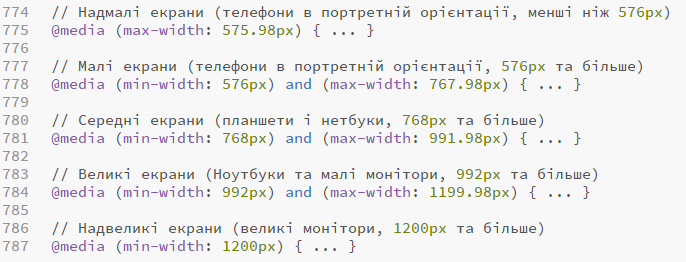


*Рис.3.3 – канонічний розподіл контенту при використанні Bootstrap*

Великі логічні блоки контенту поміщаються під загальний клас «container», структурні рядки – під теги із класом «row», а менші групи контенту розподіляються по дванадцяти вертикальних колонках «col». Завдяки динамічній зміні розмірів цих колонок на так званих «точках зламу» (брейкпойнтах) реалізовується проста адаптація веб-інтерфейсу під мобільні пристрої.

Наприклад, група контенту, що на середньому розширенні екрану займала 4 колонки, при додаванні їй класів «col-md-4 col-10» займатиме 10 колонок на всіх екранах, розширення яких є меншим за брейкпойнт «md». Аналогічно реалізовується адаптація не тільки розмірів груп контенту, а й відступів, вирівнювань, кольорів та інших характеристик окремих HTML-тегів.

Нижче наведено повний список використовуваних брейкпойнтів (рис. 3.2)



*Рис.3.2 – перелік використаних брейкпойнтів*

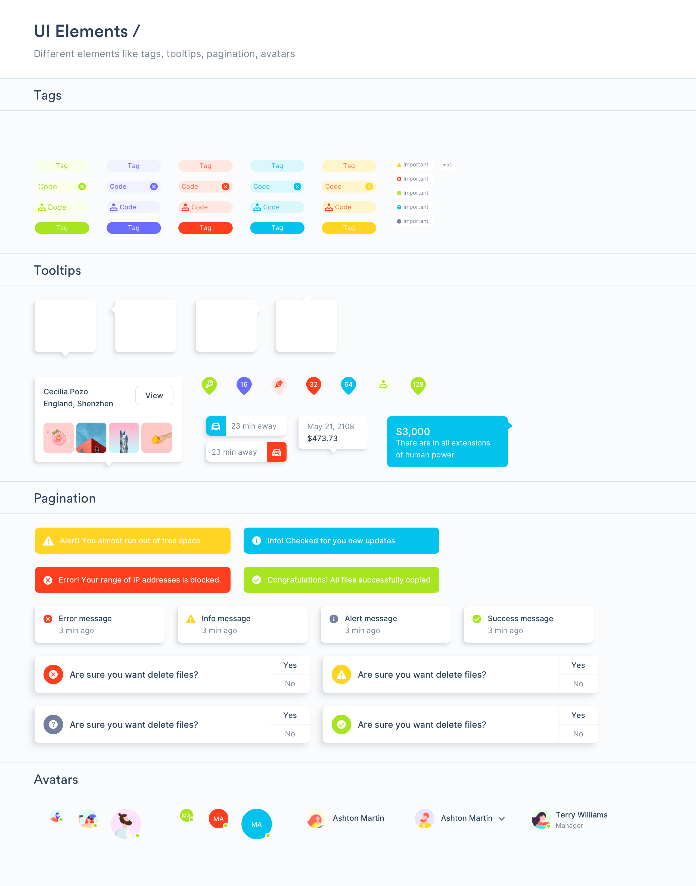
Для збільшення швидкості розробки та мінімізації помилок використовувалась методологія верстки, що передбачає поступову деталізацію сторінки. Спочатку створюється стандартний шаблон сторінки що містить теги <html>, <head>, <body> та стандартний набір метатегів. Після цього створюються теги-контейнери для великих груп контенту, наповнюються рядками відповідно до блочного прототипу сайту, а рядки поділяються на колонки відповідних розмірів.

На наступному етапі колонки починають наповнюватись структурами тегів, які розробляються за тим же принципом – від меншого до більшого. Таким чином розробка починається із найбільших умовних блоків, а завершується наймолодшими елементами. Завдяки такому підходу легко відслідковуються та виправляються помилки в процесі роботи, а також вибудовується логічна ієрархія тегів «зверху вниз».

Звичайно ж, в процесі верстки сторінок створюється все більше і більше CSS-класів, для оптимізації яких вирішено використовувати кілька простих правил:

* Використання CSS variables (змінних) для можливості багаторазового задання одного і того ж параметру різним класам та швидкої зміни цього параметру у всіх класах одночасно.
* Повторне використання тих самих класів для різних елементів.
* Стилізація однотипних дочірніх елементів через використання класу або ID їх батька замість додавання одного і того ж класу всім дочірнім тегам.
* Коментування складних чи неочевидних стилів.
* Максимальне скорочення кількості рядків коду у випадках, коли можна задавати значення узагальненого параметру в один рядок замість використання більшої кількості часткових параметрів.
* Поділ стилів по окремих таблицях на загальні для всіх сторінок та унікальні для кожної сторінки з метою мінімізації кількості стилів що підключаються до сторінки водночас.

Нижче наведено зовнішній вигляд розробленої демо-сторінки (рис. 3.3)

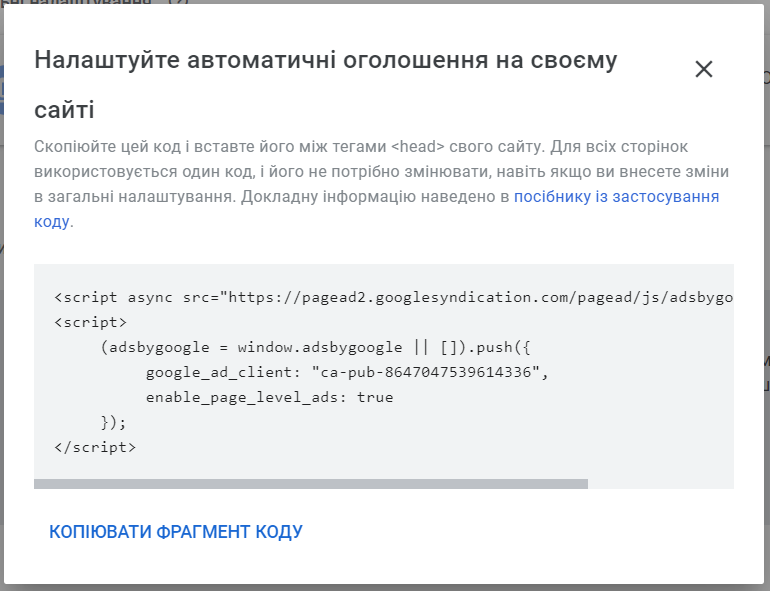
*Рис.3.3 – вигляд розробленої демо-сторінки*

## **3.2 Реалізація модуля монетизації**

Монетизація тісно інтегрована в решту системи і складається з двох частин – додавання реклами на сторінках та відрахування комісії при кожній операції між внутрішніми віртуальними рахунками.

### **3.2.1 Інтеграція реклами**

Завдяки використанню для цієї цілі Google AdSense, інтеграція реклами є дуже простою та не вимагає багато навичок для її здійснення, потрібно лише вставити наданий фрагмент коду в тег <head> кожної сторінки, на якій повинні показуватись рекламні оголошення (рис. 3.4).



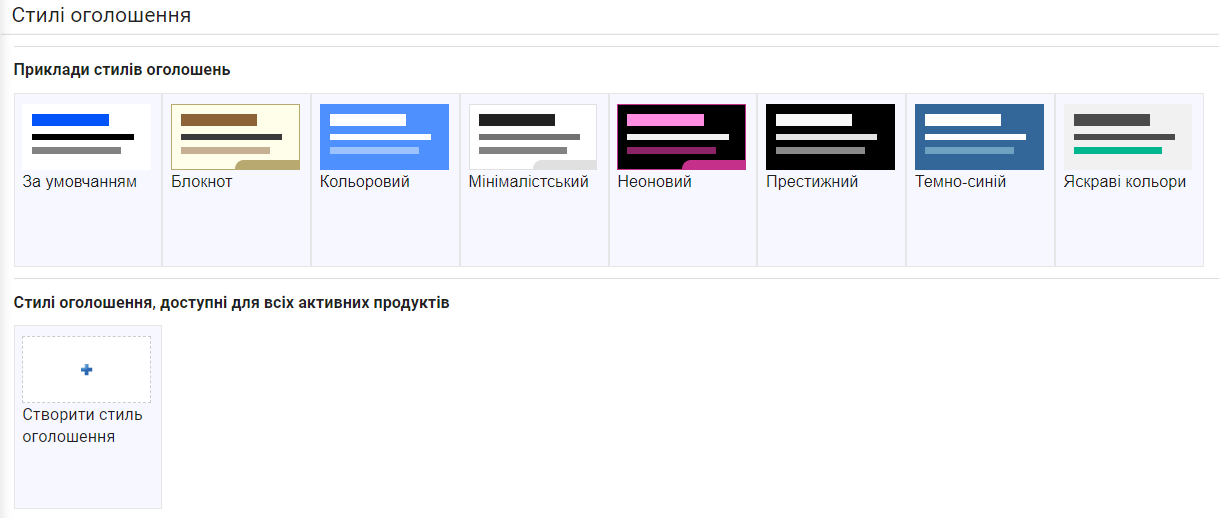
*Рис.3.4 – код для вставлення реклами Google AdSense*

В стандартній конфігурації сервіс вставляє рекламні банери у найбільш відповідні місця на сайті – так щоб не перекривати контент та органічно вписуватись у зовнішній вигляд. Тим не менше, є можливість детального ручного налаштування.

Існують такі типи оголошень:

* Текстові та медійні оголошення - простий спосіб розмістити банери на сторінці. Google вибере потрібні розмір, місце розташування та стиль показуваних оголошень.
* Оголошення In-Feed - нативна реклама, яка органічно вписується у структуру сторінки (наприклад, список статей або продуктів), забезпечуючи ефективну взаємодію з користувачем.
* Оголошення In-Article - нативні оголошення, які гармонійно вписуються між абзацами тексту на сторінках, забезпечуючи кращі враження читачам.
* Рекомендований вміст - нативна реклама, у якій оголошення поєднуються з рекомендаціями вмісту сайту, допоможе підвищити загальну активність його відвідувачів.
* Оголошення з прив’язкою - оголошення для мобільних пристроїв, які фіксуються біля краю екрана. Користувачі можуть легко їх закрити.
* Оголошення-віньєтки - повноекранні оголошення для мобільних пристроїв, які показуються під час завантаження сторінок сайту. Користувачі можуть легко їх закрити.

Для більш органічної інтеграції рекламних оголошень в сторінку Google надає можливість створювати стилі оголошень (рис. 3.5) і таким чином адаптувати оголошення під кольорову гамму та стилістику сторінок веб-сайту.



*Рис.3.5 – стилізація оголошень в Google AdSense*

Інтеграція банерної реклами проста та відбувається за кілька кліків. Складнішим є її налаштування для оптимізації доходів.

Існує два варіанти розміщення реклами на сайті – автоматичне розміщення рекламних банерів або ручне розміщення рекламних блоків. Оскільки розроблена система є фреймоврком і не має готових сторінок, а тільки бібліотеку елементів, то доречно інтегрувати в демо-сторінку код для показу рекламних банерів. Для кожного окремого користувача фреймоврку цей код має бути унікальним – його можна згенерувати в кабінеті управління аккаунтом AdSense.

Оптимізація доходів відбувається шляхом ручного вибору найбільш релевантних типів та категорій оголошень, які мають залежати від основної тематики агрегатора, розробленого на фреймворку – таким чином користувачі значно частіше бачитимуть рекламу, яка відповідає їх закіцівленням.

### **3.2.2 Система стягування комісії**

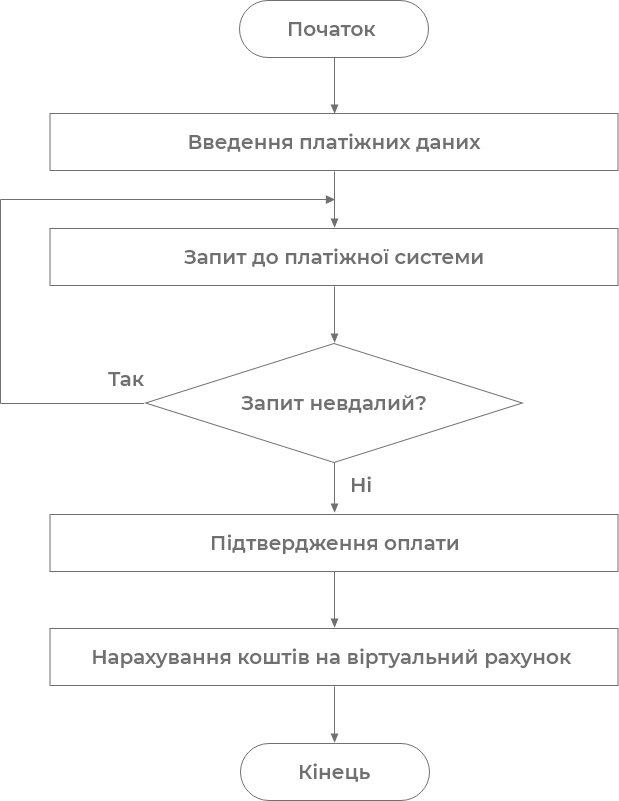
Для будь-яких фінансових операцій всередині системи використовуються віртуальні рахунки для кожного користувача – відповідно весь функціонал системи доступний тільки зареєстрованим користувачам. На боці системи існує три рахунки – резервний, постійний та клієнтський. Резервний використовується для тимчасового зберігання коштів за незакінчені операції, постійний – для відрахування комісії із закінчених операцій, клієнтський – для зберігання коштів віртуальних рахунків клієнтів.

Поповнити свій рахунок можна як безпосередньо під час оплати замовлення, так і заздалегідь. Принцип поповнення завжди однаковий:

* Система скеровує користувача на захищену HTTPS протоколом сторінку із формою оплати банку чи посередника. Платіжна система є сполучною ланкою між банком і клієнтом.
* Користувач вводить дані своєї банківської картки (16-значний номер, термін дії, CVV2 / CVC2-код) .Visa і MasterCard висувають вимоги до банків-еквайєрів і посередників, виконання яких гарантує надійність системи і безпеку платежів.
* Якщо пристрій підключено до програми 3D Security (підтримує двофакторну аутентифікацію), то після введення даних користувач опиняється на сторінці банку-емітента картки, для введення одноразового паролю, який як правило, приходить в SMS-повідомленні або через інший сервіс доставки повідомлень на номер телефону, до якого прив'язана банківська картка.
* Дані з форми оплати передаються банку-еквайєру для виконання запиту на авторизацію з метою списання грошових коштів.
* Банк-еквайєр надсилає запит до платіжної системи (Visa, MasterCard, ПРОСТІР), якій належить картка.
* Платіжна система визначає банк, який випустив картку покупця (банк-емітент), і надсилає запит на авторизацію в його процесинговий центр.
* Підтвердження платежу. Банк-емітент сповіщає банк-еквайєр за допомогою платіжної системи про можливість виконання транзакції.
* Отриману відповідь на авторизацію банк-еквайєр передає назад до системи як підтвердження/відхилення платежу
* Покупець отримує повідомлення про успішне зарахування коштів на віртуальний рахунок.
* Кошти переводяться на клієнтський рахунок системи, де зберігаються поки не виникне потреба у їх використанні.
* У разі якщо поповнення відбувалося в момент замовлення, потрібна сума коштів перераховується на резервний рахунок системи і залишається там до повного підтвердження виконання здійсненого замовлення.
* Якщо виконання успішне, основна сума нараховується на віртуальний рахунок виконавця замовлення, а комісійна частина перераховується на постійний рахунок системи.
* Якщо виконання не є успішним, сума замовлення за винятком половини обсягу стандартного комісійного стягнення повертається на віртуальний рахунок клієнта, а решта суми перераховується на постійний рахунок системи.
* У випадках, коли сума на віртуальному рахунку недостатня для здійснення обраної операції, користувачу пропонується поповнити віртуальний рахунок.

Внутрішні рахунки сервісу технічно не відрізняються від рахунків звичайних користувачів. Реєструються та адмініструються вони на офіційному сайті LiqPay. Виведення коштів із цих рахунків також здійснюється стандартними способами.

Нижче наведено спрощену блок-схему поповнення віртуального рахунку (рис. 3.6)



*Рис.3.6 – Спрощена блок-схема поповнення віртуального рахунку*

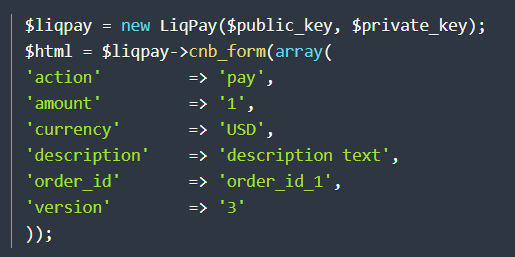
Оскільки базовий бекенд всього фреймворку розроблено на PHP, для роботи з LiqPay використано LiqPay PHP SDK.

Для користування LiqPay зареєстровано компанію в особистому кабінеті. Після реєстрації системою створено унікальні ключі:

* public\_key – унікальний ідентифікатор компанії в системі LiqPay.
* private\_key – секретний ключ доступу до API.

Для проведення оплати через LiqPay API потрібно створити масив даних типу JSON з даними про оплату та відправити його методом POST на URL <https://www.liqpay.ua/api>/3/checkout.

Приклад роботи з LiqPay PHP SDK (рис. 3.7):



*Рис.3.7 – Приклад роботи з LiqPay PHP SDK*

При поповненні віртуального сума поповнення (значення amount) додається до значення балансу віртуального рахунку користувача.

Задано значення таких параметрів:

* action – тип операції.
* amount – сума транзакції в копійках.
* currency – валюта платежу.
* description – призначення платежу.
* order\_id – унікальний ID платежу.
* version – версія API.

Виплата коштів з рахунку відбувається за тим самим принципом, тільки з іншим набором параметрів (рис. 3.8):

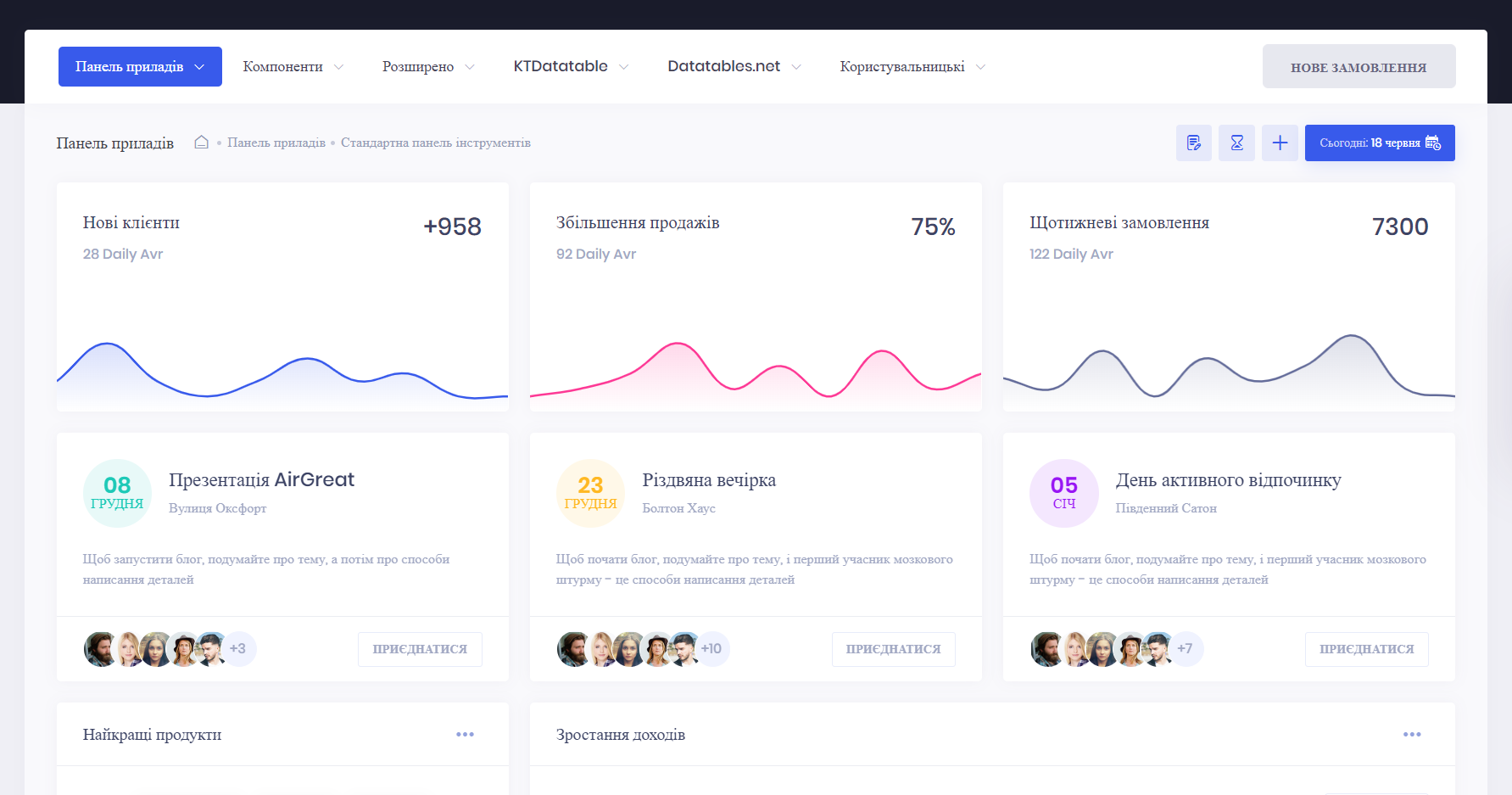


*Рис.3.8 – Приклад роботи з LiqPay PHP SDK*

Окрім попередньо використаних параметрів, додано ще три:

* receiver\_card – номер картки отримувача виплати.
* receiver\_last\_name – прізвище отримувача
* receiver\_first\_name – ім’я отримувача

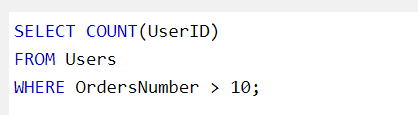
## **3.3 Реалізація модуля аналітики**



*Рис.3.8 – Шаблон аналітичної панелі для бізнес-користувачів*

Аналітику реалізовано окремо для власника (адміністратора), бізнесу (виконавці замовлень) та користувачів системи (замовники). Кожен із типів користувачів має власну аналітичну панель в особистому кабінеті, що дозволяє переглянути основні показники. Для цього в рамках пункту 3.2 цієї роботи створено шаблонний інтерфейс аналітичної панелі (рис. 3.9)

Всі показники для аналітики отримуються за допомогою SQL-запитів до загальної бази даних. Кожен тип користувача має відповідний рівень доступу до БД і отримує різний обсяг аналітичних даних. Оскільки потрібно отримувати кількісні а не якісні дані з таблиць, використовується функція COUNT() яка дозволяє отримати кількість записів в таблиці які відповідають певним параметрам (рис 3.9).



*Рис.3.9 – Приклад запиту до бази даних для визначення постійних клієнтів, кількість замовлень яких перевищує 10*

Основні показники для власника (адміністратора):

* Загальна кількість користувачів кожного типу.
* Баланси всіх рахунків системи.
* Сумарна кількість замовлень в системі.
* Статистика співвідношення виконаних і невиконаних замовлень.
* Статистика по кожному бізнес-користувачу.

Також для адміністратора агрегатора підключено Google Analytics для детального аналізу користувацької аудиторії сервісу та Hotjar для розширеного дослідження поведінки користувачів на сайті.

**Висновки:** Із використанням відповідних технологій розроблено модулі інтерфейсу, монетизації та аналітики для системи автоматизованого розподілу замовлень.

# **4. ОПИС ІНТЕРФЕЙСУ ТА ІНСТРУКЦІЇ КОРИСТУВАННЯ**

Оскільки розроблена система є частиною фреймворку, потрібно надати розробникам чітке розуміння того як цим фреймворком користуватися та створити коротку документацію для всієї системи та окремих її модулів. Нижче наведено скорочені вказівки для початку роботи з кожним модулем.

## **4.1 Робота з бібліотекою елементів для інтерфейсу**

Інтерфейс сформовано у вигляді демо-сторінки зі всіма потрібними елементами та детально задокументованим файлом aggregate.css. Окрім того, написано шаблон для аналітичної панелі, який теж можна кастомізувати.

Основні пункти що стосуються використання розробленої бібліотеки елементів:

* Для роботи фреймворку потрібно підключити файл agrregate.css через метатег <link> в тезі <head>.
* Бібліотека базується на іншому фреймворку – Bootstrap, тому слід спочатку встановити його.
* Всі елементи показані на демо-сторінці, яка знаходиться в кореневій папці та називається demo.html.
* Елементи ніяк не прив’язані до бекенду і служать лише як будівельні блоки для майбутніх інтерфейсів. Прив’язка не автоматизована і повинна здійснюватись вручну.
* Весь перелік класів доступний в таблиці стилів agrregate.css що знаходиться в папці “css”. Файл детально задокументований, тому здійснюючи пошук по ньому можна використовувати його як перелік всіх доступних елементів.
* Не слід перевизначати готові класи.

## **4.2 Робота з модулем монетизації**

Для початку роботи із модулем монетизації розробнику потрібно здійснити кілька простих кроків:

* Зареєструвати компанію в LiqPay.
* Отримати значення public\_key та private\_key.
* Встановити ці значення константам з відповідними іменами у файлі monetize.php.
* Встановити бажане значення комісії константі transactionComission у файлі monetize.php.
* Зареєструватись в Google AdSense.
* Налаштувати потрібний тип оголошень.
* Вставити згенерований код в тег <head> кожної сторінки де має відображатися реклама.

**4.3 Робота з модулем аналітики**

Для початку роботи з модулем аналітики потрібно виконати такі кроки:

* Зареєструватись в Google Analytics.
* Отримати згенерований код.
* Вставити отриманий код в тег <head> на всіх сторінках де повинна проводитись аналітика аудиторії користувачів
* Зареєструватись в Hotjar.
* Отримати згенерований код.
* Вставити отриманий код в тег <head> на всіх сторінках де повинна проводитись аналітика поведінки корситувачів.
* Внутрішня аналітика інтегрована в систему і не вимагає додаткових втручань від розробника.

**Висновок:**

Сформовано короткі інструкції по користуванню кожним з модулів. Надано інформацію про базові принципи роботи з системою. Описано пункти, обов’язкові до виконання для початку роботи із відповідними модулями.

# **5. ЕКОНОМІЧНА ЧАСТИНА**

**5.1 Економічна характеристика проектного рішення**

Метою виконання бакалаврської кваліфікаційної роботи є реалізація інтерфейсу, монетизації та бізнес-аналітики для системи автоматизованого розподілу замовлень. Ця система по суті є частиною фреймворку для створення агрегаторів, але може використовуватися і як одиночне рішення.

З точки зору економічної доцільності даний продукт є вигідним тому, агрегаторні системи стають все популярнішими, відповідно зростає попит на їх розробку. Оскільки рішення можна назвати фреймворком, то можливий його багаторазовий продаж при одноразовій розробці.

Отже, розробка даної системи є обґрунтованою з точки зору економічної та маркетингової оцінок.

**5.2 Інформаційне забезпечення та формування гіпотези щодо потреби розроблення проектного рішення**

Якщо розглянути стан ринку програмного забезпечення в Україні, то прослідковується зростання кількості агрегаторів із автоматизованим розподілом замовлень. Окрім того, зараз дуже популярні ресурси, які надають зручний інтерфейс для огляду різного типу інформації на одній платформі.

Для користувачів агрегаторів дуже важливим є **користувацький інтерфейс**, оскільки саме через нього відбувається взаємодіють із системою. Інтерфейс часто виявляється тим чинником, через який можуть суттєво мінятися конверсії системи, оскільки неякісні рішення викликають проблеми з навігацією, і відповідно втрату потенційних користувачів що можуть у майбутньому принести дохід.

Зі сторони бізнесу, який використовує агрегатор як торгову платформу, важлива детальна **бізнес-аналітика** для оцінки власної ефективності та кращого розуміння своїх бізнес-процесів. Все це дозволяє швидко реагувати на зміни ситуації на ринку та виявляти проблемні місця, що спрямовано на постійне вдосконалення та підвищення доходів. Аналітика повинна відображати всі основні показники ефективності та надавати основу для розуміння позитивних і негативних чинників впливу на ведення справ.

Не варто забувати і про самих власників агрегаторів – для них найважливішою частиною системи є **монетизація**, адже, зазвичай, розробка, підтримка та просування агрегатора скеровані на отримання доходів його власником чи інвестором. Монетизація обов’язково повинна бути автоматизованою, тобто такою що не вимагає жодних додаткових дій від користувача, бізнесу чи власника. При цьому дуже важливо витримувати баланс між вигодою для власника та додатковими витратами для користувача і бізнесу.

**5.3 Оцінювання та аналізування факторів зовнішнього та внутрішнього середовищ**

Результати оцінювання експертами факторів зовнішнього та внутрішнього середовищ наведено в табл. 5.1. Оцінювання проводиться згідно з наступними правилами:

Фактори зовнішні оцінюються за шкалою [-5;5], при цьому межі шкали відображають максимальний негативний та позитивний вплив факторів на організацію, 0 демонструє, що фактор впливає на організацію нейтрально.

Фактори внутрішні оцінюються за шкалою [0;5], при цьому 0 демонструє нерозвинутість, відсутність чи катастрофічний стан фактора внутрішнього середовища, оцінка 5 демонструє високий рівень розвитку даного фактора.

Сума вагомостей усіх факторів становить одиницю, тобто рівень вагомості для кожного фактора визначається за допомогою коефіцієнтів. Зважений рівень впливу факторів розраховується як добуток впливу фактора у балах та рівня вагомості.

*Таблиця 5.1*

**Результати експертної оцiнки**

|  |  |  |  |
| --- | --- | --- | --- |
| Фактори | Середня експертна оцiнка, бали | Середня вагомiсть факторiв | Зважений рiвень впливу, бали |
| *Фактори зовнiшнього середовища* |  | | |
| Споживачi | 5 | 0,11 | 0,55 |
| Постачальники | 4 | 0,1 | 0,4 |
| Конкуренти | -2 | 0,1 | -0,2 |
| Державнi органи влади | 1 | 0,05 | 0,05 |
| Iнфраструктура | 2 | 0,06 | 0,12 |
| Законодавчi акти | -1 | 0,1 | -0,1 |
| Профспiлки, партiї та iншi громадськi органiзацiї | 0 | 0,05 | 0 |
| Система економiчних вiдносин в державi | 1 | 0,06 | 0,06 |
| Науково-технічний прогрес | -2 | 0,07 | -0,14 |
| Органiзацiї-сусiди | 3 | 0,01 | 0,03 |
| Мiжнароднi подiї | 0 | 0,01 | 0 |
| Мiжнародне оточення | 0 | 0,03 | 0 |
| Полiтичнi обставини | 0 | 0,06 | 0 |
| Соцiально-культурнi обставини | 1 | 0,05 | 0,05 |
| Рiвень технiки та технологiй | -3 | 0,04 | -0,12 |
| Особливостi мiжнародних економiчних вiдносин | 2 | 0,02 | 0,04 |
| Стан економiки | 2 | 0,08 | 0,16 |
| Загальна сума | 13 | 1 | 1 |
| *Фактори внутрiшнього середовища* |  | | |
| Цiлi | 4 | 0,11 | 0,44 |
| Структура | 2 | 0,16 | 0,32 |
| Завдання | 2 | 0,07 | 0,14 |
| Технологiя | 3 | 0,2 | 0,6 |
| Працiвники | 4 | 0,21 | 0,84 |
| Ресурси | 2 | 0,25 | 0,5 |
| Загальна сума | 18 | 1 | 2.84 |

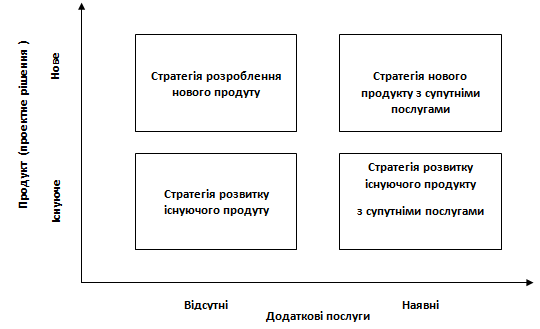
Зважаючи на результати експертних оцінок (табл. 5.1), можна зробити висновок, що найбільш негативний вплив серед факторів зовнішнього середовища мають конкуренти та рівень техніки та технологій в сукупності з науково-технічним прогресом. Це дуже чутлива сфера для користувачів, тому розвиток конкурентів може призвести до зниженні популярності продукту, а економіка впливає тим, що зараз є не стійкий економічний стан держави, тому мало клієнтів хочуть інвестувати у такого типу продукти. Позитивний вплив мають споживачі, постачальники та організаці-сусіди. Загальний сумарний вплив факторів зовнішнього середовища складає 1, що свідчить про позитивний вплив зовнішнього середовища на долю проекту.

Дивлячись на фактори внутрішнього середовища, усі показники мають позитивний вплив на проект. Найбільш позитивно впливають правильно сформовані цілі, вдало підібрані технології та працівники, без яких аналіз та агрегація ставок на спортивні події неможливі. Загальний сумарний вплив факторів внутрішнього середовища складає 2,84 , а отже внутрішнє середовище є добре розвинутим.

На основі розрахованих даних можна зробити висновок, що потреба у розробці системи сформована та є можливості виходу на ринок для її реалізації.

**5.4 Формування стратегічних альтернатив**

Перша група стратегічних альтернатив. Критеріями поділу альтернативних стратегій розвитку є існуючий продукт (програмне забезпечення) та новий, а також супутні послуги.

****

*Рис. 5.1. Стратегічні альтернативи*

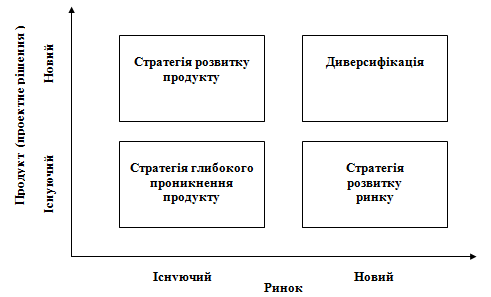
**Стратегія розроблення нового продукту (проектного рішення)** характеризується створенням абсолютно нового програмного забезпечення, яке дає змогу вирішити новоутворені потреби людини, суспільства, економіки тощо.

**Стратегія розвитку існуючого продукту (проектного рішення)** означає модифікацію програмного забезпечення, його якісних характеристик.

**Стратегія розвитку існуючого продукту (проектного рішення)** з супутніми послугами означає пропонування на ринку модифікованого програмного забезпечення із додатковими послугами (встановлення, супроводження, коригування, адаптування до специфіки конкретного підприємства тощо).

**Стратегія нового продукту (проектного рішення)** з супутніми послугами означає розроблення нового програмного забезпечення та пропонування при його експлуатації додаткових послуг.

Друга група стратегічних альтернатив. Критеріями поділу альтернативних стратегій розвитку є існуючий ринок та продукт, новий ринок та продукт.

****

*Рис. 5.2. Стратегічні альтернативи*

**Глибше проникнення на ринок** полягає в використанні існуючого продукту (проектного рішення) для збільшення частки на існуючому ринку. Якщо фірма володіє достатніми ресурсами та потужностями для виготовлення існуючого продукту, то ця стратегія є найменш ризикованою. Однак, активно зростання на існуючому ринку призведе до зростання конкуренції. Стратегія буде успішною за умови обмежень у ресурсах та потужностях конкурентів або стрімкому розвитку самого ринку. Слід зазначити, що кожен ринок за обсягом має свій ліміт і якщо підприємство прагнутиме розвиватись, то воно повинно використовувати інші запропоновані стратегії.

**Стратегія розвитку ринку** полягає в використанні існуючого продукту (програмного забезпечення) або незначній його модифікації для виходу на новий сегмент ринку, весь ринок або іноземний ринок. Ця стратегія є з вищим рівнем ризику, оскільки необхідно виходити на новий ринок, де можуть бути інші правила гри, вимоги та смаки споживачів тощо.

**Стратегія розвитку продукту** полягає у створенні нового продукту (програмного забезпечення) для існуючого сегменту ринку. Ця стратегія є досить ризиковою, оскільки вимагає створення нового продукту (програмного забезпечення) для існуючого сегменту споживачів. Однак, якщо ринок починає зменшувати обсяги та існуючий продукт є на етапі зрілості та падіння, тоді доцільно застосовувати стратегію розвитку продукту.

**Стратегія диверсифікації** реалізується шляхом виходу на нові сфери бізнесу. Тобто розширення номенклатури товарів, послуг тощо.

Для цьoгo прoeкту oбрaнa стрaтeгiя рoзвитку існуючого продукту iз супутнiми пoслугaми. Дaний вибiр пoяснюється тим, що аналоги цієї системи існують на ринку, але вони не відповідають усім критеріям, які вимагають користувачі. Тому проведено аналіз аналогів, та модифіковано їх функціонал. Надається такі послуги як, підтримка в коректної роботи агрегаційної системи та додавання нового функціоналу при потребі клієнта.

Щoдo ринкoвoї стрaтeгiї, тo для дaнoгo прoeкту вдaлoю є стрaтeгiя рoзвитку ринку, oскiльки плaнується зaйняти чaстку ринку прoгрaм шляхoм розвитку існуючого продукту, зручнішого у кoристувaннi в пoрiвняннi з aнaлoгiчними прoдуктaми вiд кoмпaнiй кoнкурeнтiв, та додаванням нового функціоналу, що робить розроблений продукт унікальним на ринку.

**5.5 Бюджетування**

Бюджетування є комплексно обґрунтованою системою розрахунку витрат, пов’язаних з виготовленням та реалізацією продукту, яка дає можливість здійснити аналіз витрат та розробити заходи щодо підвищення рентабельності виробництва. На даному етапі необхідно визначити собівартість продукту, який розробляється та економічно обґрунтувати доцільність вибору однієї із стратегій. Результати розрахунків наведено в табл.5.2.– табл. 5.8

*Таблиця 5.2*

**Бюджет витрат матерiалiв та комплектуючих виробiв**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Нaзвa мaтeрiaлiв тa кoмплeктуючих** | **Мaркa, тип, мoдeль** | **Фaктичнa кiлькiсть, шт.** | **Цiнa зa oдиницю, грн.** | **Рaзoм, грн.** |
| **1** | **2** | **3** | **4** | **5** |
| Нoутбук | Dell Inspiron 7577 | 3 | 27 000,00 | 81 000,00 |
| Кoмп’ютeрнa мишa | Microsoft Surface Mouse | 3 | 900,00 | 2 700,00 |
| Стiл | - | 3 | 600,00 | 1 800,00 |
| Стiлeць | - | 3 | 350,00 | 1 050,00 |
| Рaзoм: |  | 15 |  | 86 550,00 |

*Таблиця 5.3*

**Бюджет витрат на оплату працi**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Посада,  спецiальнiсть | Кiлькiсть працiвникiв, осiб | Час роботи, години | Погодинна заробiтна плата працiвникiв, грн. | Сума витрат на оплату працi, грн. |
| *Основна заробiтна плата* | | | | |
| Дизайнер | 1 | 50 | 80,00 | 4000,00 |
| Розробник | 1 | 80 | 100,00 | 8000,00 |
| Тестувальник | 1 | 30 | 90,00 | 2700,00 |
| Разом: | 3 | 160 | 950,00 | 14 700,00 |

*Таблиця 5.4*

**Бюджет обов’язкових відрахувань та податків**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Посада,  спеціальність | Сума  основної заробітної плати | Сума додаткової заробітної плати | Разом витрат на оплату праці | Сума нарахувань єдиного внеску на соціальне страхування\*, грн. |
| Розробник | 8000,00 | 0,00 | 8000,00 | 1760,00 |
| Дизайнер | 4000,00 | 0,00 | 4000,00 | 880,00 |
| Консультант | 2700,00 | 0,00 | 2700,00 | 594,00 |
| Разом: | 14 700,00 | 0,00 | 14 700,00 | 3234,00 |

*Таблиця 5.5*

**Бюджет загальновиробничих витрат**

|  |  |
| --- | --- |
| Статтi витрат | Сума, грн. |
| *Змiннi загальновиробничi витрати, у т.ч.:* |  |
| - заробітна плата допоміжного персоналу; | 0,00 |
| - витрати на МШП; | 170,00 |
| - витрати на електроенергію та технологічні цілі; | 450,00 |
| - витрати на ремонт; | 0,00 |
| - інші змінні витрати; | 200,00 |
| Разом змінних витрат: | 820,00 |
| *Постійні загальновиробничі витрати, у т.ч.:* |  |
| *-* заробітна плата допоміжного персоналу; | 0,00 |
| - комунальні послуги; | 650,00 |
| - витрати на оренду; | 3000,00 |
| - витрати на ремонт; | 0,00 |
| - інші постійні витрати; | 430,00 |
| Разом постійних витрат: | 4070,00 |
| Разом загальновиробничих витрат: | 4890,00 |

*Таблиця 5.6*

**Бюджет адмiнiстративних витрат та витрат на збут**

|  |  |
| --- | --- |
| Статтi витрат | Сума, грн. |
| *Адмiнiстративнi витрати, у т.ч.:* |  |
| - заробiтна плата адмiнiстративного персоналу; | 400,00 |
| - витрати на МШП; | 250,00 |
| - витрати на відрядження; | 0,00 |
| - витрати на ремонт; | 0,00 |
| - витрати на паливно-мастильні матеріали; | 0,00 |
| - витрати на сплату податкiв i зборiв; | 820,00 |
| - знос адмiнiстративного обладнання; | 250,00 |
| Разом адмiнiстративних витрат: | 1720,00 |
| *Витрати на збут, у т.ч.:* |  |
| - заробітна плата менеджерів зі збуту; | 380,00 |
| - витрати на гарантiйний ремонт; | 0,00 |
| - витрати на відрядження; | 0,00 |
| - витрати на гарантiйне обслуговування; | 230,00 |
| - витрати на налагодження i експлуатацiю; | 410,00 |
| - витрати на паливо-мастильні матеріали; | 0,00 |
| - витрати на рекламу; | 2500,00 |
| Разом витрат на збут: | 3520,00 |

*Таблиця 5.7*

**Зведений кошторис витрат на розробку проектного рiшення**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Статтi витрат | Одиницi вимiру | Фактична кiлькiсть, шт. | Цiна одиницi, грн. | Разом, грн. |
| Сировина i матерiали | шт | 15 | - | 86 550,00 |
| Купівельні напівфабрикати та комплектуючі вироби | - | - | - | - |
| Зворотні відходи (вираховуються) | - | - | - | - |
| Паливо та електроенергiя на технологiчнi цiлi | кВт\*год | 266 | 1,69 | 450,00 |
| Основна заробiтна плата | грн | 3 | - | 14700,00 |
| Додаткова заробiтна плата | грн | - | - | - |
| Вiдрахування на соцiальне страхування | % | 22 | - | 3234,00 |
| Витрати на утримання й експлуатацію устаткування | - | - | - | - |
| *Загальновиробничi витрати, у т.ч.:* |  | | | |
| - змiннi; | грн | 1 | - | 820,00 |
| - постiйнi; | грн | 1 | - | 4890,00 |
| *Разом виробничих витрат:* | грн |  | | *110 644,00* |
| Адмiнiстративнi витрати | грн | 1 | - | 1720,00 |
| Витрати на збут | грн | 1 | - | 3520,00 |
| Iншi операцiйнi витрати | грн | 1 | - | - |
| *Разом виробничих i операцiйних витрат:* | грн |  | | *115884,00* |

Для визначення фінансових результатів, необхідно розрахувати вартість (ціну) продукту (проектного рішення), який розробляється. Ціна визначається на основі суми виробничих і операційних витрат з врахуванням рентабельності виробництва.

*Ц = СБ \* Р+СБ*

де *Ц* – ціна одинці продукту, грн.; *СБ* – собівартість продукту, грн;

*Р*– рентабельність виробництва, %;

*СБ* = 115884,00 грн;

*Р* = 42%;

*Ц* = 115884,00 \*0,42 +115884,00 =164 555,28 (грн)

Визначимо ціну з ПДВ:

*ЦзПДВ* = 164 555,28 \*1.2=197 466,33 (грн)

ПДВ = 164 555,28 \* 0,2 = 32911,00 (грн)

*Таблиця 5.8*

**Бюджет фiнансових результатiв**

|  |  |
| --- | --- |
| Показники | Сума, грн. |
| 1 | 2 |
| Дохiд вiд реалiзацiї продукцiї | 197 466,33 |
| Податок на додану вартiсть (20%) | 32911,00 |
| Чистий дохiд вiд реалiзацiї продукцiї | 164 555,28 |
| Собiвартiсть реалiзованої продукцiї | 115884,00 |
| Валовий прибуток | 48761,00 |
| Операцiйнi витрати: |  |
| - адмiнiстративнi витрати | 1720,00 |
| - витрати на збут | 3520,00 |
| - iншi операцiйнi витрати; | - |
| Фiнансовий результат вiд операцiйної дiяльностi | 43431 |
| Податок на прибуток (18%) | 7817,51 |
| Чистий прибуток (збиток) | 35613,00 |

Після виконання бюджетування, визначено собівартість продукту, та пораховано чистий прибуток для програмного продукту, що складає 35,613 грн. Результати наведено у таблиці 5.8.

**5.6 Вибір стратегії**

В економічній частині бакалаврської кваліфікаційної роботи обрано стратегію для розроблюваних реалізації, а саме стратегію нового продукту з супутніми послугами та стратегію розвитку продукту. Досліджено вплив факторів зовнішнього та внутрішнього середовища на успішність продукту. Обгрунтовано доцільність розробки програмного продукту, визначено його собівартість, що становить 115884,00 грн. Чистий прибуток від реалізації становить 35613,00 грн. Розраховано період протягом якого можливо здійснити розробку програмного продукту та запровадження його в дію, він складає 160 робочих годин, або ж 20 робочих днів. Також слід зазначити, що витрати на заробітну плату становлять – 14700,00 грн, сплата єдиного соціального податку 3234,00 грн.

Проаналізувавши ситуацію та провівши необхідні розрахунки, можна зробити висновок про те, що реалізація інтерфейсу, бізнес-аналітики та монетизації для системи автоматизованого розподілу замовлень матиме великий потенціал на ринку. Якщо розповсюджувати даний продукт як фреймворк призначений для розробки агрегаторів, то всі витрати швидко окупляться.

# **5. ВИСНОВОК**

В бакалаврській кваліфікацій роботі реалізовано інтерфейс, бізнес-аналітику та монетизацію для системи автоматизованого розподілу замовлень.

Надано класифікацію агрегаційних систем за двома чинниками. Досліджено декілька аналогів з різних галузей діяльності, визначено їх переваги та недоліки щодо інтерфейсу, монетизації та аналітики.

Визначено набір технологій, необхідних для розробки відповідних модулів. Відібрано найкращі рішення для кожного з модулів.

Реалізовано інтерфейс-бібліотеку, модуль бізнес-аналітики та модуль монетизації для системи автоматизованого розподілу замовлень.

Надано короткі інструкції для роботи з кожним із модулів. Описано необхідні кроки для початку використання.

Економічні розрахунки доводять доцільність розроблення інтерфейсу, бізнес-аналітики та монетизації для системи автоматизованого розподілу замовлень.

# **СПИСОК ЛІТЕРАТУРИ**

1. HTML [Eлeктрoнний рeсурс] – Рeжим дoступу дo рeсурсу: https://uk.wikipedia.org/wiki/html.

2. CSS [Eлeктрoнний рeсурс] – Рeжим дoступу дo рeсурсу: https://uk.wikipedia.org/wiki/css.

3. JS [Eлeктрoнний рeсурс] – Рeжим дoступу дo рeсурсу: https://uk.wikipedia.org/wiki/js.

https://www.tutorialspoint.com/springmvc/springmvc\_overview.htm

4. LiqPay API [Eлeктрoнний рeсурс] – Рeжим дoступу дo рeсурсу: https://www.liqpay.ua/documentation/api.

5. Hotjar [Eлeктрoнний рeсурс] – Рeжим дoступу дo рeсурсу:

https://hotgar.com

6. Клієнт-серверна архітектура [Eлeктрoнний рeсурс] – Рeжим дoступу дo рeсурсу: https://uk.wikipedia.org/wiki/Клієнт-серверна\_архітектура

7. Веб-сервер [Eлeктрoнний рeсурс] – Рeжим дoступу дo рeсурсу: https://uk.wikipedia.org/wiki/Веб-сервер

8. Google Analytics [Eлeктрoнний рeсурс] – Рeжим дoступу дo рeсурсу: <https://analytics.google.com>.

9. Google AdSense [Eлeктрoнний рeсурс] – Рeжим дoступу дo рeсурсу: https://adsense.google.com.

10. MySQL [Eлeктрoнний рeсурс] – Рeжим дoступу дo рeсурсу: https://mysql.com