

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики
Кафедра комп'ютерних наук та програмної інженерії

Розробка лендінгу факультету комп'ютерних наук, фізики та
математики для стейкхолдерів

(назва теми)

Кваліфікаційна робота (проект)
на здобуття ступеня вищої освіти “бакалавр”

Виконав: здобувач 4 курсу 12-441 групи

Спеціальності 121 Інженерія програмного
забезпечення

(шифр, назва)

Освітньо-професійної програми
Інженерія програмного забезпечення

(назва)

Рудчук Максим Олегович

(ПІБ)

Керівник доктор фізико-математичних
наук, професор Песчаненко Володимир
Сергійович

(наук. ступінь, вчене звання, П.І.Б.)

Рецензент к.т.н., доцент Захарченко Раїса
Миколаївна

(наук. ступінь, вчене звання, П.І.Б.)

ЗМІСТ

ВСТУП	3
РОЗДІЛ 1. ТЕОРЕТИЧНІ ТА ПРОЄКТНІ ЗАСАДИ РОЗРОБКИ ЛЕНДИНГУ ФАКУЛЬТЕТУ ЗВО	6
1.1. Лендинг як інструмент комунікації ЗВО з абітурієнтами	6
1.2. Інформаційні потреби абітурієнтів та UX-вимоги до лендингу факультету ІТ-спрямування	9
1.3. Аналіз структурних рішень лендингів українських ІТ-шкіл	15
1.4. Технологічний стек класичної веб-розробки: огляд компонентів та критерії вибору	19
1.5. AI-інструменти як допоміжний засіб у сучасній веб-розробці	25
Висновки до Розділу 1	30
РОЗДІЛ 2. РОЗРОБКА ТА РЕАЛІЗАЦІЯ ЛЕНДИНГУ ФАКУЛЬТЕТУ	31
2.1. Проєктування лендингу: інформаційна архітектура, контент, дизайн- система	31
2.2. Реалізація фронтенду: верстка, адаптивність, інтерактивні елементи	38
2.3. Реалізація серверної частини на Laravel	43
2.4. Адміністративна панель на Laravel Orchid	48
2.5. Контейнеризація та розгортання	53
2.6. Застосування AI-інструментів у процесі розробки: спостереження та практичні висновки	58
Висновки до Розділу 2	63
ВИСНОВКИ	68
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	72
ДОДАТКИ	75

ВСТУП

Актуальність теми. В умовах війни російської федерації проти України заклади вищої освіти, що були змушені до евакуації, постали перед якісно новим завданням комунікації з абітурієнтами. Херсонський державний університет, основний кампус якого розташований у тимчасово окупованому Херсоні, з 2022 року здійснює освітню діяльність у дистанційному форматі з тимчасовим перебуванням в Івано-Франківську. У таких умовах веб-присутність факультетів університету перестає бути допоміжним каналом інформування та перетворюється на основний інструмент залучення вступників.

Факультет комп'ютерних наук, фізики та математики ХДУ готує фахівців у галузях, де очікування абітурієнтів щодо якості веб-комунікації об'єктивно вищі, ніж у середньому для університетських підрозділів. Потенційний вступник на спеціальності «Інженерія програмного забезпечення», «Комп'ютерні науки» чи «Інформаційні системи та технології» сприймає сам сайт факультету як непрямий показник рівня підготовки. Водночас на момент початку роботи факультет не мав окремого веб-ресурсу і був представлений лише сторінкою у складі загального сайту університету, що обмежувало можливості цілеспрямованої комунікації з абітурієнтами.

Окремий лендинг як спеціалізована цільова сторінка, орієнтована на конкретний сегмент аудиторії, відповідає сучасній практиці провідних закордонних університетів і дозволяє вирішити вказану проблему. Розробка такого ресурсу для факультету комп'ютерних наук, фізики та математики ХДУ є практичним завданням, що поєднує проєктування інформаційної архітектури, дизайн інтерфейсу, реалізацію клієнтської та серверної частин, побудову адміністративного інструментарію та розгортання продукту в умовах реальної експлуатації.

Мета дослідження — розробити та розгорнути функціональний лендинг факультету комп'ютерних наук, фізики та математики Херсонського державного університету, орієнтований на потреби абітурієнтів як ключової групи стейкхолдерів, із застосуванням сучасного інженерного стеку та допоміжних AI-інструментів на окремих етапах розробки.

Завдання дослідження:

1. Проаналізувати поняття лендингу як інструменту веб-комунікації ЗВО, визначити інформаційні потреби абітурієнтів факультету IT-спрямування та узагальнити структурні рішення лендингів закордонних факультетів комп'ютерних наук.
2. Обґрунтувати вибір технологічного стеку для реалізації лендингу та розглянути роль сучасних AI-інструментів як допоміжного засобу у веб-розробці.
3. Спроекувати інформаційну архітектуру, дизайн-систему та структуру бази даних лендингу відповідно до виявлених потреб цільової аудиторії.
4. Реалізувати клієнтську частину, серверну частину та адміністративну панель лендингу із забезпеченням можливості автономного управління контентом представниками факультету.
5. Здійснити контейнеризацію та розгортання продукту в умовах реальної експлуатації, описати застосування AI-інструментів на етапах розробки та сформулювати практичні спостереження щодо їх використання.

Об'єкт дослідження — процес розробки веб-ресурсу типу лендинг для структурного підрозділу закладу вищої освіти.

Предмет дослідження — технологічні, проєктні та організаційні рішення при розробці лендингу факультету комп'ютерних наук, фізики та математики ХДУ.

Практичне значення одержаних результатів. Результатом роботи є функціональний лендинг факультету комп'ютерних наук, фізики та математики ХДУ, розгорнутий за адресою fknfm.kspu.edu та переданий в експлуатацію факультету. Ресурс містить інформацію про сім бакалаврських та п'ять магістерських освітніх програм, реалізованих у межах шести спеціальностей факультету, опис інфраструктури дистанційного навчання, представлення викладацького складу, інформацію для українських та іноземних абітурієнтів, форму зворотного зв'язку та інші структурні елементи. Адміністративна панель забезпечує можливість самостійного оновлення контенту представниками факультету без залучення розробника. Зафіксований у роботі досвід застосування AI-інструментів на етапах проєктування та розробки може становити практичний інтерес для студентів та розробників-початківців.

Структура роботи. Кваліфікаційна робота складається зі вступу, двох розділів, висновків, списку використаних джерел та додатків. У першому розділі розглянуто теоретичні засади розробки лендингів для ЗВО, проаналізовано інформаційні потреби абітурієнтів, узагальнено структурні рішення закордонних аналогів, обґрунтовано вибір технологічного стеку та розглянуто роль AI-інструментів у сучасній веб-розробці. У другому розділі описано процес проєктування та реалізації лендингу, включно з розробкою фронтенду, серверної частини, адміністративної панелі та розгортанням, а також узагальнено практичний досвід використання AI-інструментів на різних етапах роботи. Список використаних джерел містить 27 найменувань. Робота містить 11 таблиць та 4 додатки.

РОЗДІЛ 1

ТЕОРЕТИЧНІ ТА ПРОЄКТНІ ЗАСАДИ РОЗРОБКИ ЛЕНДИНГУ ФАКУЛЬТЕТУ ЗВО

1.1. Лендинг як інструмент комунікації ЗВО з абітурієнтами

Перш ніж переходити до проєктування конкретного веб-ресурсу, необхідно зафіксувати, що саме розуміється під поняттям «лендинг» у межах цієї роботи, у чому полягає його функціональна відмінність від інших типів веб-ресурсів та яку роль такий ресурс може виконувати в комунікації закладу вищої освіти з абітурієнтами.

Поняття лендингу. Термін *landing page* (англ. «цільова сторінка») увійшов у практику веб-розробки наприкінці 1990-х років у контексті інтернет-маркетингу і початково позначав сторінку, на яку потрапляв користувач після переходу за рекламним посиланням. У сучасному вжитку поняття суттєво розширилося: лендингом прийнято називати веб-сторінку, спроектовану з конкретною комунікативною метою та орієнтовану на конкретну дію відвідувача — заповнення форми, реєстрацію, перехід до іншого ресурсу або ознайомлення з певним набором інформації. На відміну від багатосторінкового сайту, що поєднує функції довідника, новинного видання та сервісного інструменту для різних категорій користувачів, лендинг концентрує увагу однієї цільової групи на обмеженій кількості ключових повідомлень [5].

Сучасна література з веб-юзабіліті та конверсійної оптимізації виділяє кілька характеристик, що відрізняють лендинг від інших типів веб-ресурсів:

- **односторінкова або мінімальна структура** з послідовним розгортанням контенту у вертикальному напрямку;

- **орієнтація на конкретну цільову дію** (*call to action*), що визначає композицію сторінки та ієрархію її елементів;
- **обмежена навігація**, що знижує ризик відволікання уваги відвідувача на нерелевантні розділи;
- **сегментованість аудиторії** — лендинг проєктується під чітко окреслену групу користувачів, інформаційні потреби яких відомі або реконструйовані на етапі дослідження.

Зазначені характеристики не є самоціллю, а впливають з функціонального призначення лендингу: за обмежений час перебування користувача на сторінці донести ключове повідомлення та спонукати до запланованої дії. Це відрізняє лендинг від, наприклад, корпоративного сайту, де одна сторінка одночасно адресована вкрай різномірним групам — клієнтам, партнерам, працівникам, медіа.

Лендинг у контексті діяльності ЗВО. Сайт закладу вищої освіти як комунікативний ресурс має складну структуру аудиторій. Відповідно до концепції стейкхолдерів, сформульованої у класичних роботах з теорії стратегічного управління [8], до зацікавлених сторін діяльності ЗВО належать здобувачі освіти, абітурієнти, науково-педагогічні працівники, роботодавці, органи державної влади, грантодавці, академічні партнери, випускники та інші групи. Кожна з цих груп має власні інформаційні потреби, мову опису і поведінкові патерни. У спробі обслужити всі категорії одночасно офіційний сайт університету неминуче перетворюється на ресурс компромісного характеру: інформація, релевантна абітурієнтові, опиняється поряд із інформацією, що становить інтерес для випускників, працівників приймальної комісії або міжнародних партнерів. Це знижує ефективність такого сайту як інструменту залучення нових вступників.

Лендинг факультету розв'язує цю проблему через виокремлення цільової аудиторії та проєктування інформаційної архітектури під її потреби. Замість універсального ресурсу, що балансує між інтересами

різних груп, створюється спеціалізований інструмент, орієнтований передусім на абітурієнтів — тобто на групу, від поведінки якої безпосередньо залежить набір факультету і, відповідно, його подальша діяльність.

Слід зазначити, що використання терміну «лендинг» стосовно факультетського веб-ресурсу не є канонічним і може викликати заперечення з боку прихильників вужчого, маркетингового тлумачення поняття. Така позиція має раціональні підстави: класичний лендинг у комерційному контексті оптимізується під одну метрику — конверсію відвідувача у клієнта — і часто є частиною ширшої рекламної кампанії [5]. Факультетський веб-ресурс таких жорстких метрик зазвичай не має. Водночас за основними структурними та функціональними ознаками — односторінковою архітектурою, сегментованістю аудиторії, наявністю чіткого СТА, орієнтацією на конкретну дію (звернення до приймальної комісії, ознайомлення з освітніми програмами) — факультетський веб-ресурс відповідає характеристикам лендингу як типу веб-сторінки. У межах цієї роботи термін використовується саме в такому розширеному значенні, що склалося в практиці веб-розробки.

Українська практика та можливості розвитку. Більшість українських ЗВО на сьогодні обмежуються представленням окремих факультетів у межах загального сайту університету — у вигляді розділу або сторінки з обмеженою функціональністю та контентом, поданим у форматі офіційної інформації. Це дозволяє забезпечити базове інформаційне покриття, проте недостатньо враховує специфіку абітурієнта як цільової аудиторії: спосіб подання інформації, навігаційні рішення, візуальне оформлення зазвичай орієнтовані на узагальненого користувача, а не на потенційного вступника віком 16–18 років. Окремий лендинг факультету, спроектований під цю групу, може суттєво підвищити ефективність комунікації — як завдяки структурі

контенту, так і завдяки тональності і візуальній мові, відповідній очікуванням аудиторії.

Особливого значення такий підхід набуває для факультетів ІТ-спрямування, де професійні стандарти власної галузі формують у потенційних вступників відповідні очікування й щодо якості веб-представництва. Сайт, побудований за застарілими шаблонами або з очевидними помилками у проєктуванні, для цієї аудиторії стає непрямым аргументом проти вибору відповідного закладу. Цей чинник зумовлює актуальність розробки окремого, технічно та візуально якісного лендингу для факультету комп'ютерних наук, фізики та математики ХДУ — як завдання, що поєднує комунікативну, маркетингову та технологічну складові.

Висновок до підрозділу. Лендинг у межах цієї роботи розглядається як спеціалізований веб-ресурс із односторінковою архітектурою, спроектований під конкретну групу стейкхолдерів — у цьому випадку абітурієнтів факультету — з метою цілеспрямованої комунікації та спонукання до визначеної дії (звернення до приймальної комісії, ознайомлення з освітніми програмами, заповнення форми зворотного зв'язку). Подальший виклад побудовано таким чином: у підрозділі 1.2 деталізовано інформаційні потреби абітурієнтів ІТ-факультету, у підрозділі 1.3 узагальнено структурні рішення закордонних аналогів, у підрозділах 1.4–1.5 розглянуто технологічну базу реалізації.

1.2. Інформаційні потреби абітурієнтів та UX-вимоги до лендингу факультету ІТ-спрямування

Ефективність лендингу як інструменту комунікації визначається тим, наскільки повно він відповідає інформаційним потребам та поведінковим очікуванням цільової аудиторії. Для лендингу факультету такою аудиторією є абітурієнти — особи, які перебувають у процесі

вибору закладу вищої освіти та освітньої програми. У цьому підрозділі реконструюються основні групи їх інформаційних потреб і відповідні вимоги до інтерфейсу веб-ресурсу.

Абітурієнт як цільова аудиторія. Український абітурієнт бакалаврського рівня — це, як правило, особа віком 16–18 років, випускник закладу загальної середньої освіти, який обирає спеціальність для подальшого навчання у ЗВО. Магістерський рівень охоплює іншу вікову категорію — переважно випускників бакалаврських програм віком 21–22 роки, у частині випадків — осіб з досвідом роботи, що повертаються до навчання задля переорієнтації або поглиблення кваліфікації. Обидві групи об'єднує спільна риса: вони належать до поколінь, що сформувалися в середовищі високої цифрової насиченості, мають усталені практики споживання онлайн-контенту і використовують мобільні пристрої як основний канал доступу до інформації.

Прийняття рішення про вступ до ЗВО є багатоетапним процесом, що включає усвідомлення потреби в освіті, пошук інформації про потенційні заклади, оцінювання альтернатив, формування короткого списку та подання документів [4]. На етапі пошуку інформації веб-ресурси закладів є одним із ключових джерел поряд з рекомендаціями знайомих та відкритими рейтингами. Це визначає функцію лендингу як інструменту, що працює на стадіях пошуку та оцінювання альтернатив, формуючи перше системне враження про факультет та надаючи інформацію, достатню для зваженого рішення.

Групи інформаційних потреб. На підставі аналізу типових сценаріїв поведінки абітурієнта та вимог нормативного регулювання вищої освіти в Україні можна виокремити чотири основні групи інформаційних потреб, які має задовольняти лендинг факультету.

Перша група — інформація про освітні програми та спеціальності. Абітурієнт потребує чіткого розуміння переліку

доступних програм, їх змісту, тривалості, форми навчання та кваліфікації, що присвоюється після завершення. Для факультету комп'ютерних наук, фізики та математики ХДУ це передбачає подання інформації про сім бакалаврських та п'ять магістерських освітніх програм у межах шести спеціальностей: «Інженерія програмного забезпечення», «Комп'ютерні науки», «Інформаційні системи та технології», «Фізика та астрономія», а також спеціальності зі сфери освіти за напрямками математики, фізики та інформатики. Особливе значення має інформація про перелік ключових дисциплін та професійні перспективи випускників, оскільки саме ці параметри найчастіше визначають вибір на користь конкретної програми [3].

Друга група — умови вступу та фінансові аспекти. Сюди належать перелік необхідних документів, терміни подачі заяв, вступні випробування, вартість контрактного навчання, наявність бюджетних місць та інформація про мінімальні бали для зарахування. Право абітурієнта на отримання повної та достовірної інформації про умови вступу закріплене Законом України «Про вищу освіту» [1] і визначає мінімальний обсяг інформації, яку зобов'язаний містити веб-ресурс закладу.

Третя група — навчальне середовище та інфраструктура. Сучасні абітурієнти цікавляться не лише академічним змістом програм, а й тим, у якому середовищі їм доведеться навчатися. Це охоплює інформацію про матеріально-технічну базу факультету, інструменти дистанційного навчання, можливості участі в наукових проєктах і програмах міжнародної мобільності, а також формат організації освітнього процесу. Для ХДУ, що здійснює освітню діяльність переважно в дистанційному форматі через тимчасову окупацію основного кампусу, ця група потреб набуває особливого значення: абітурієнт має отримати чітке уявлення про те, як саме буде

організоване його навчання та які цифрові інструменти він використовуватиме.

Четверта група — комунікація з факультетом. Незалежно від обсягу інформації, поданої на сторінці, у абітурієнта неминуче залишаються індивідуальні питання, для відповіді на які потрібен контакт із представниками приймальної комісії або деканату. Це визначає необхідність наявності контактних даних у структурно помітному місці лендингу, а також форми зворотного зв'язку, що дозволяє залишити запит без переходу на зовнішні ресурси.

Поведінкові особливості та вимоги до UX. Окрім самого складу інформації, проєктування лендингу має враховувати спосіб, у який цільова аудиторія сприймає онлайн-контент. Це формує низку вимог до користувацького інтерфейсу, частина яких є універсальними для будь-якого сучасного веб-ресурсу, а частина — специфічними для аудиторії абітурієнтів.

Адаптивність (responsive design) є обов'язковою умовою з огляду на переважне використання мобільних пристроїв для доступу до інтернету молодіжною аудиторією. Сайт має зберігати функціональність та читабельність на екранах різного розміру — від мобільних телефонів до повноформатних моніторів — без втрати ключової інформації або порушення композиції [11].

Швидкість завантаження безпосередньо впливає на готовність користувача залишатись на сторінці. Тривале очікування відображення першого змістовного контенту є однією з найчастіших причин відмови від подальшого ознайомлення з ресурсом. Це визначає необхідність оптимізації обсягу графічних матеріалів, ефективної завантаженості скриптів та правильної організації архітектури сторінки.

Структурованість викладу. Зважаючи на коротку тривалість первинного огляду сторінки, інформація має бути організована таким чином, щоб ключові повідомлення сприймалися за лічені секунди. Це

досягається використанням заголовків і підзаголовків з чіткою ієрархією, виокремленням ключових тверджень, застосуванням іконок та інфографіки замість суцільного текстового опису, а також логічним групуванням споріднених елементів [10].

Помітність та зрозумілість елементів заклику до дії. Кнопки, що ведуть до подачі заявки, отримання консультації, ознайомлення з програмами, повинні бути візуально виокремлені та однозначно описувати свою функцію. Невизначений або багатозначний СТА знижує ймовірність переходу до запланованої дії, навіть якщо абітурієнт у принципі готовий її виконати.

Доступність (web accessibility, A11y). Веб-доступність — це сукупність вимог до проєктування інтерфейсів, що забезпечують можливість використання ресурсу людьми з різними формами обмежень: порушеннями зору, слуху, моторики, когнітивними особливостями. Базові принципи доступності сформульовано у міжнародному стандарті WCAG (Web Content Accessibility Guidelines) [27], що визначає чотири засадничі вимоги: контент має бути сприйнятним (perceivable), функціональним (operable), зрозумілим (understandable) та надійним для використання різними технологіями (robust). На практичному рівні це означає необхідність семантичної HTML-розмітки (використання тегів <header>, <nav>, <main>, <section> замість універсального <div>), наявності текстових альтернатив для зображень (атрибут alt), достатнього контрасту тексту і фону, можливості навігації клавіатурою без використання миші, коректної роботи з програмами екранного читання (screen readers). Для лендингу факультету ЗВО доступність має не лише етичний, а й прагматичний аспект: серед потенційних абітурієнтів неминуче присутні особи з функціональними обмеженнями, для яких сайт із грубими порушеннями A11y буде фактично недоступним. Окремо слід відзначити, що значна частина рішень, які забезпечують доступність

(семантична розмітка, осмислені alt-тексти, послідовна структура заголовків), одночасно покращують SEO-показники ресурсу, оскільки пошукові системи інтерпретують структуру сторінки за тими ж принципами, що й допоміжні технології.

Зазначені вимоги корелюють із загальними принципами юзабіліті, сформульованими у класичних роботах із проєктування користувацьких інтерфейсів [10; 12]: легкість засвоєння інтерфейсу, ефективність використання при повторних взаємодіях, мінімізація помилок, естетика та узгодженість оформлення. Дотримання цих принципів є необхідною, хоча й недостатньою умовою створення ефективного лендингу.

Зведена відповідність потреб і структурних елементів.

Виокремлені групи інформаційних потреб становлять основу для проєктування структури лендингу, що детально розглядається у Розділі 2 цієї роботи. Узагальнену відповідність між групами потреб та секціями лендингу подано в таблиці 1.1.

Таблиця 1.1

Відповідність інформаційних потреб абітурієнтів структурним секціям лендингу

Група інформаційних потреб	Секція лендингу
Формування первинного враження про факультет, ключова ціннісна пропозиція	Него-секція; секція «Про факультет»
Інформація про освітні програми, спеціальності та професійні перспективи	Секція «Освітні програми»; секція «Кар'єрні перспективи»
Інформація про навчальне середовище та формат організації освітнього процесу	Секція «Екосистема дистанційного навчання»; секція «Переваги факультету»
Умови вступу, терміни, документи, відповіді на типові питання	Секція «Для абітурієнтів»; секція FAQ
Комунікація з факультетом та приймальною комісією	Секція «Викладачі»; контактна секція; форма зворотного зв'язку

Послідовність розташування секцій відповідає природному порядку сприйняття: від загального уявлення про факультет — через інформацію про освітні можливості — до конкретних кроків стосовно вступу.

Висновок до підрозділу. Інформаційні потреби абітурієнтів ІТ-факультету охоплюють чотири основні групи: освітні програми, умови вступу, навчальне середовище та канали комунікації. Поведінкові особливості цільової аудиторії визначають вимоги до адаптивності, швидкодії, структурованості викладу, помітності елементів заклику до дії та доступності веб-ресурсу. Сукупність цих вимог становить основу для проєктування структури лендингу, реалізованого в межах цієї роботи. Перш ніж переходити до проєктування, доцільно розглянути практику побудови подібних ресурсів у провідних закордонних університетах — цьому присвячено наступний підрозділ.

1.3. Аналіз структурних рішень лендингів українських ІТ-шкіл

Як було зазначено у підрозділі 1.1, концепція лендингу як маркетингово-орієнтованої цільової сторінки найповніше реалізується у комерційному сегменті освітніх продуктів. Провідні закордонні факультети комп'ютерних наук — Stanford CS, MIT EECS, ETH Zürich — мають веб-представництво у форматі повноцінних багатосторінкових сайтів з розгорнутою структурою розділів для різних категорій користувачів. Водночас в українському освітньому ринку існує сегмент, у якому лендинг є основним інструментом залучення аудиторії — приватні ІТ-школи, що конкурують за того ж абітурієнта, який є цільовою аудиторією факультету комп'ютерних наук, фізики та математики ХДУ.

Обґрунтування вибірки. Для аналізу обрано трьох найбільших гравців українського ринку ІТ-освіти: Mate Academy [23], GoIT [19] та Hillel IT School [21]. Вибір зумовлений такими міркуваннями. По-перше,

всі три організації існують у формі онлайн-шкіл з масштабованою аудиторією і використовують лендинг як основний інструмент конверсії відвідувача у студента. По-друге, цільова аудиторія цих шкіл значно перетинається з аудиторією бакалаврських програм ІТ-спрямування українських ЗВО: випускники середньої школи та особи, які розглядають ІТ як напрям професійної переорієнтації. По-третє, на відміну від університетських сайтів, ці лендинги оптимізовані під вимірювані метрики — заявку, реєстрацію, замовлення консультації — а отже відображають апробовані практики проєктування саме лендингу як типу веб-ресурсу [5]. Розглядати їх як прямих конкурентів ЗВО було б некоректно — освітні продукти суттєво відрізняються за змістом, тривалістю і статусом отриманого диплома, — проте з точки зору структурних рішень лендингу вони становлять обґрунтовану референтну вибірку.

Об'єкти аналізу. Mate Academy — українська онлайн-школа, що пропонує курси Front-end, Full-stack, Java, Python, QA, UI/UX, DevOps та інших спеціальностей з моделлю відкладеної оплати після працевлаштування. GoIT — школа з 2014 року роботи, з власною LMS-платформою та широким спектром програм у галузі програмування, дизайну, аналітики, тестування. Hillel IT School — одна з провідних ІТ-шкіл України, що пропонує курси з програмування, тестування, дизайну та маркетингу, із сертифікаційними іспитами ISTQB та орієнтацією на ринок праці. Усі три організації мають окремі лендинги головної сторінки та окремі лендинги під конкретні курси, що дозволяє проаналізувати як загальні, так і деталізовані рішення.

Узагальнені структурні елементи. Аналіз публічно доступних лендингів зазначених шкіл (стан на травень 2026 р.) дозволяє виокремити набір структурних елементів, що є спільними для всіх трьох ресурсів і становлять, фактично, типову анатомію сучасного освітнього лендингу. Узагальнення наведено в таблиці 1.2.

Структурні елементи лендингів провідних українських ІТ-шкіл (за станом на травень 2026 р.)

Структурний елемент	Mate Academy	GoIT	Hillel
Hero-секція з ключовою ціннісною пропозицією	+	+	+
Чіткий заклик до дії (запис на курс, консультація)	+	+	+
Каталог освітніх програм / курсів	+	+	+
Опис цільових професійних результатів	+	+	+
Інформація про формат та тривалість навчання	+	+	+
Орієнтовна вартість або модель оплати	+	+	+
Соціальний доказ: відгуки, історії випускників	+	+	+
Інформація про викладачів / менторів	+	+	+
Партнерські компанії	+	+	+
FAQ / відповіді на типові запитання	+	+	+
Форма зворотного зв'язку / заявки на консультацію	+	+	+
Адаптивний дизайн (mobile-first)	+	+	+

Спільні структурні рішення. На основі таблиці можна виокремити елементи, наявні у всіх трьох ресурсах і, відповідно, такі, що становлять стандарт для лендингів освітнього спрямування в українському ринку:

- hero-секція з ключовою ціннісною пропозицією та основним візуальним акцентом;
- чіткий заклик до дії, орієнтований на отримання заявки чи консультації;

- структурований каталог освітніх продуктів з можливістю переходу до деталізованої сторінки;
- експліцитна прив'язка освітнього продукту до професійних результатів (посада, орієнтовний рівень доходу, кар'єрні перспективи);
- соціальний доказ у вигляді відгуків випускників та інформації про компанії-роботодавці;
- FAQ-секція як механізм проактивної відповіді на типові заперечення;
- форма зворотного зв'язку як основна точка конверсії.

Особливості, що відрізняють розглянуті ресурси від лендингу факультету ЗВО. Поряд із загальними структурними рішеннями важливо зафіксувати чинники, через які пряме копіювання підходу комерційних ІТ-шкіл у контексті ЗВО було б методологічно некоректним. По-перше, ІТ-школи оптимізують лендинг під одну прозору метрику — заявку, а через неї — оплачений курс. Лендинг факультету ЗВО функціонує у складнішій системі регуляторних обмежень (правила прийому, ЗНО/НМТ, ЄВІ/ЄФВВ, терміни вступної кампанії), і пряме спонукання до «купівлі» тут неможливе. По-друге, тональність маркетингової комунікації ІТ-шкіл — наприклад, акцент на швидкому досягненні високого доходу — у ЗВО є недоречною: тут йдеться не лише про професійну підготовку, а й про здобуття академічного ступеня з усіма пов'язаними формальними процедурами. По-третє, ІТ-школи позиціонують освітній продукт як шлях до конкретної посади (Junior Frontend, QA, UX/UI Designer), тоді як університетська бакалаврська програма формує ширшу професійну базу, яка може реалізовуватись у різних напрямках.

Зазначене не знецінює структурного аналізу — навпаки, воно вказує, які саме елементи доцільно перенести у факультетський лендинг (структура, навігація, FAQ, форма зворотного зв'язку, представлення

викладачів, інформація про програми), а які — переосмислити з урахуванням специфіки ЗВО (тональність повідомлень, формулювання ціннісної пропозиції, характер закликів до дії).

Висновок до підрозділу. Аналіз лендингів трьох провідних українських IT-шкіл — Mate Academy, GoIT та Hillel IT School — дозволив виокремити дванадцять типових структурних елементів сучасного освітнього лендингу, з яких сім становлять стійкий стандарт для всіх розглянутих ресурсів. Ці елементи покладені в основу проєктування структури лендингу факультету комп'ютерних наук, фізики та математики ХДУ, представленого у Розділі 2. Водночас перенесення цих рішень здійснюється з урахуванням специфіки факультетського веб-ресурсу як інструменту комунікації академічного закладу, що накладає змістові та тональні обмеження, відмінні від комерційного освітнього сегменту.

1.4. Технологічний стек класичної веб-розробки: огляд компонентів та критерії вибору

Класична веб-розробка передбачає послідовне використання набору технологій, кожна з яких відповідає за конкретний шар застосунку: представлення інформації користувачеві, обробку запитів на сервері, зберігання даних, управління контентом та розгортання продукту в експлуатаційному середовищі. У цьому підрозділі розглянуто склад технологічного стеку, обраного для реалізації лендингу факультету комп'ютерних наук, фізики та математики ХДУ, та обґрунтовано рішення щодо кожного компонента.

Критерії вибору технологій. При формуванні стеку для проєктів освітнього сектору доцільно враховувати чотири основні групи критеріїв.

Зрілість і стабільність. Університетський веб-ресурс має експлуатуватися протягом тривалого часу із мінімальним технічним

обслуговуванням. Це визначає перевагу технологій з усталеним статусом у галузі, передбачуваним циклом релізів та підтримкою з боку постачальника або відкритої спільноти.

Доступність підтримки. Університет як замовник, на відміну від комерційної організації, не має постійного штату веб-розробників. Це означає, що подальша підтримка ресурсу — оновлення контенту, точкові доопрацювання, виправлення помилок — може здійснюватися різними особами в різний час, у тому числі студентами та аспірантами. Стек має ґрунтуватися на широко відомих технологіях, документація яких легко доступна, а спільнота — численна.

Можливість автономного управління контентом. Веб-ресурс факультету має передбачати самостійне оновлення інформації представниками деканату без залучення розробника. Це визначає необхідність повноцінної адміністративної панелі, інтегрованої з основним застосунком, а не побудованої окремим інструментом.

Переносність та відтворюваність розгортання. Технологічна інфраструктура університету може змінюватися — від виділеного сервера до хмарного провайдера, від однієї операційної системи до іншої. Стек має забезпечувати мінімально болісне перенесення продукту між середовищами без переписування конфігурацій.

З урахуванням цих критеріїв сформовано стек, склад якого детально розглядається далі.

Компоненти стеку. У таблиці 1.3 наведено перелік основних компонентів технологічного стеку лендингу із зазначенням їх версій (станом на травень 2026 р.) та функціонального призначення в архітектурі застосунку.

Таблиця 1.3

Технологічний стек лендингу факультету

Компонент	Версія	Призначення в архітектурі
-----------	--------	---------------------------

Компонент	Версія	Призначення в архітектурі
PHP	8.2+	Серверна мова програмування
Laravel	12	PHP-фреймворк для побудови серверної частини
Orchid Platform	14.52+	Пакет для побудови адміністративної панелі
MySQL	8+	Реляційна СУБД для зберігання контенту
Apache	2.4+	HTTP-сервер для обслуговування запитів
Blade	вбудований у Laravel	Шаблонізатор для серверного рендерингу HTML
Astrotomic Translatable	11.16+	Бібліотека для двомовного контенту
Vite	5+	Інструмент збірки фронтенд-ресурсів
HTML5, CSS3, JavaScript	актуальні стандарти	Клієнтська частина
Docker, Docker Compose	актуальні	Контейнеризація та оркестрація сервісів
Git	2+	Система контролю версій

Далі розглянуто ключові компоненти стеку з обґрунтуванням вибору щодо кожного з них.

Серверна частина: PHP та Laravel. Серверна частина побудована на мові програмування PHP версії 8.2 із використанням фреймворку Laravel [22]. PHP залишається однією з найпоширеніших серверних мов у сегменті веб-розробки і має широку базу інфраструктурної підтримки з боку хостинг-провайдерів, що особливо актуально для проєктів академічного сектору з обмеженими ресурсами на DevOps. Версія 8.2 запровадила суттєві типобезпекові поліпшення, включаючи readonly-властивості, що позитивно впливає на якість серверного коду.

Laravel як фреймворк обрано з кількох причин. По-перше, він реалізує архітектурний патерн MVC, що забезпечує чітке розділення відповідальності між шарами застосунку — моделлю даних,

контролерами та представленнями [14]. По-друге, Laravel містить вбудовану ORM Eloquent, систему міграцій бази даних, шаблонізатор Blade, систему маршрутизації, валідації запитів та локалізації — тобто практично всі компоненти, потрібні для типового веб-застосунку, доступні з коробки без додаткової інтеграції різномірних бібліотек. По-третє, Laravel має одну з найбільших спільнот серед PHP-фреймворків, обширну документацію та сталий цикл релізів LTS. Версія 12, актуальна на момент розробки, гарантує підтримку оновлень безпеки протягом найближчих років.

Адміністративна панель: Laravel Orchid. Однією з функціональних вимог до лендингу є можливість автономного управління контентом представниками факультету. Реалізація адміністративної панелі з нуля є об'ємним завданням, що включає розробку інтерфейсу авторизації, форм редагування для кожної сутності, таблиць перегляду даних, фільтрації, сортування та валідації. У межах цієї роботи для цієї мети обрано пакет Orchid Platform [24] — спеціалізований інструмент побудови адміністративних інтерфейсів поверх Laravel.

Orchid реалізує концепцію декларативних «екранів» (Screens) — компонентів, що описують структуру адміністративної сторінки через PHP-класи. Розробник визначає набір полів, дій, таблиць та інших UI-компонентів, а Orchid самостійно генерує відповідний HTML, обробляє користувацький введення та зв'язує його з моделями Eloquent. Такий підхід дозволяє побудувати повнофункціональну адміністративну панель з CRUD-операціями для всіх сутностей лендингу без написання клієнтського коду для адміністративної частини. Альтернативами є пакети Filament та Nova; Orchid обрано через його ліцензійну відкритість (на відміну від комерційного Nova) та відносно простішу криву входу для розробника, який вперше працює з адмін-пакетами Laravel.

База даних: MySQL. Для зберігання динамічного контенту лендингу обрано реляційну СУБД MySQL [25]. Вибір зумовлений тим, що MySQL є однією з найпоширеніших СУБД з відкритим вихідним кодом, забезпечує високу продуктивність на типових для лендингу обсягах даних, підтримується практично всіма хостинг-провайдерами та має повну сумісність з ORM Eloquent у Laravel. PostgreSQL розглядався як альтернатива і має певні переваги в роботі зі складними типами даних та повнотекстовим пошуком, проте для лендингу з обмеженим набором сутностей переваги MySQL у поширеності та простоті адміністрування переважили.

Клієнтська частина: HTML5, CSS3, JavaScript, Blade.

Клієнтська частина реалізована на стеку HTML5 + CSS3 + JavaScript [6; 9] без використання фреймворків типу React або Vue. Це рішення зумовлене характером застосунку: лендинг є переважно контентно-орієнтованим веб-ресурсом з обмеженою кількістю інтерактивних елементів (карусель у hero-секції, таби в секції програм, акордеон у FAQ, мобільне меню). Використання SPA-фреймворку для такого продукту створило б невиправдану складність архітектури, погіршило показники SEO та збільшило час первинного завантаження сторінки.

Серверний рендеринг сторінки виконується за допомогою шаблонізатора Blade, що є вбудованим компонентом Laravel. Blade забезпечує чисту інтеграцію HTML-розмітки з PHP-логікою, підтримує спадкування шаблонів та компонентний підхід через директиви `@component` та `@include`, що дозволяє організувати фронтенд за модульним принципом — кожна секція лендингу реалізується як окремий Blade-шаблон з відповідними стилями та скриптами.

Окремою вимогою до клієнтської частини є дотримання принципів web accessibility, розглянутих у підрозділі 1.2. На рівні розмітки це реалізується через використання семантичних HTML5-тегів (`<header>`, `<nav>`, `<main>`, `<section>`, `<article>`, `<footer>`),

що передають структуру сторінки програмам екранного читання та пошуковим роботам, через послідовну ієрархію заголовків від `<h1>` до `<h3>`, через атрибути `alt` для всіх змістовних зображень, через використання ARIA-атрибутів для інтерактивних компонентів, що не мають вбудованих семантичних аналогів (карусель, акордеон, кастомний випадаючий список), а також через підтримку клавіатурної навігації для всіх інтерактивних елементів [27]. Blade як шаблонізатор не накладає обмежень на дотримання цих вимог — повний контроль над генерованою розміткою лежить на розробнику.

Двомовність: `astrotoomic/laravel-translatable`. Лендинг має підтримувати дві мови — українську (основну) та англійську (для іноземних абітурієнтів). Реалізація багатомовності засобами «з нуля» вимагала б створення дублювальних полів для кожної мови у кожній таблиці, що значно ускладнило б структуру БД. Замість цього використано пакет `astrotoomic/laravel-translatable` [17], що реалізує патерн «модель + таблиця перекладів»: основна таблиця містить мовнонезалежні поля (зображення, посилання, числові значення), а текстові поля винесені в окрему таблицю перекладів, де кожен запис прив'язаний до конкретної мови. Такий підхід забезпечує чисту структуру даних та простоту додавання нових мов у майбутньому.

Збірка фронтенду: `Vite`. Для збірки клієнтських ресурсів використано `Vite` [26] — сучасний інструмент збірки, що замінив застарілий `Laravel Mix` у нових версіях фреймворку. `Vite` забезпечує швидкий dev-сервер з функцією `hot module replacement`, оптимізовану продакшн-збірку з мініфікацією та версіонуванням файлів для управління кешуванням браузера.

Контейнеризація та розгортання: `Docker`. Для забезпечення відтворюваності середовища та спрощення розгортання використано `Docker` [18] з оркестрацією через `Docker Compose`. Конфігурація описує два сервіси: контейнер застосунку (`ksu-fmf-app`), що містить PHP-

інтерпретатор разом із вбудованим веб-сервером Apache, та контейнер MySQL. Така структура є мінімально достатньою для лендингу і дозволяє розгорнути повне середовище однією командою `docker compose up` на будь-якій машині з установленим Docker, незалежно від операційної системи.

Висновок до підрозділу. Сформований технологічний стек поєднує зрілі компоненти з широкою підтримкою спільноти (PHP, MySQL, Apache, Docker) з сучасними інструментами розробки (Laravel 12, Orchid 14, Vite) і відповідає критеріям, сформульованим на початку підрозділу. Дотримання класичної архітектури — серверний рендеринг Blade-шаблонів, реляційна БД, окрема адміністративна панель — є обґрунтованим вибором для лендингу як типу веб-ресурсу і одночасно забезпечує можливість подальшої підтримки продукту широким колом потенційних виконавців.

1.5. AI-інструменти як допоміжний засіб у сучасній веб-розробці

Стрімкий розвиток технологій штучного інтелекту у 2022–2026 роках суттєво вплинув на практику розробки програмного забезпечення. Великі мовні моделі (Large Language Models, LLM), такі як GPT від OpenAI, Claude від Anthropic, Gemini від Google, продемонстрували здатність генерувати працездатний програмний код на основі описів природною мовою, пояснювати наявний код, пропонувати виправлення помилок та автоматизувати рутинні задачі [7]. Поява спеціалізованих інструментів генерації інтерфейсів, таких як Google Stitch [20], розширила цю можливість на етап проєктування дизайну. У цьому підрозділі розглянуто загальний стан AI-інструментів у веб-розробці на момент виконання цієї роботи (травень 2026 р.), їх класифікацію за функціональним призначенням, типові обмеження, а також

обґрунтовано вибір конкретних інструментів, застосованих при розробці лендингу.

Загальна класифікація AI-інструментів у веб-розробці. За функціональним призначенням сучасні AI-інструменти, що застосовуються у веб-розробці, можна умовно поділити на чотири основні категорії, узагальнення яких наведено в таблиці 1.4.

Таблиця 1.4

Категорії AI-інструментів у сучасній веб-розробці (станом на травень 2026 р.)

Категорія	Призначення	Типові представники
Універсальні LLM-асистенти	Діалогова взаємодія, генерація коду, пояснення помилок, рефакторинг, проєктування архітектури	Claude (Anthropic), ChatGPT (OpenAI), Gemini (Google)
Спеціалізовані AI-асистенти у середовищі розробки	Контекстна автодоповнюваність та генерація коду безпосередньо в IDE	GitHub Copilot, Cursor, Windsurf
Генератори інтерфейсів	Створення дизайну та фронтенд-коду на основі текстового опису або референсних зображень	Google Stitch, Vercel v0, Claude Artifacts
Платформи повного циклу	Поєднання генерації коду, попереднього перегляду, тестування та розгортання в одному середовищі	Bolt (StackBlitz), Lovable, Replit Agent

Зазначені категорії не є взаємовиключними — багато сучасних інструментів поєднують функції двох і більше категорій. Так, Claude доступний як у формі діалогового веб-інтерфейсу (універсальний асистент), так і у складі Claude Code — командного інструменту, що працює в локальному середовищі розробника. GitHub Copilot містить як автодоповнення в IDE, так і чат-режим, наближений до універсальних

LLM. Vercel v0 поєднує генерацію інтерфейсів з можливістю одразу розгортати результат.

Універсальні LLM-асистенти як основний інструмент розробника. Найширше розповсюдженою категорією AI-інструментів у веб-розробці є універсальні мовні моделі, з якими розробник взаємодіє у діалоговому форматі. Розробник формулює завдання природною мовою, надає необхідний контекст (наявний код, вимоги, опис помилки), модель генерує відповідь — фрагмент коду, пояснення, конфігурацію, документацію. Сучасні моделі підтримують значний обсяг контексту в межах однієї сесії — до сотень тисяч і навіть мільйонів токенів, — що дозволяє працювати з повноцінними кодовими базами, а не лише з ізольованими фрагментами.

Серед моделей цієї категорії на момент виконання цієї роботи Claude від Anthropic [16] вважається однією з найкращих для задач, пов'язаних із генерацією та аналізом коду. Сімейство Claude складається з трьох рівнів: Opus (максимальна якість міркувань та довгої роботи з кодом), Sonnet (баланс якості та вартості, найчастіше використовуваний у щоденній розробці) та Haiku (швидкість та низька вартість для масових простих задач). Альтернативними інструментами цієї ж категорії є GPT від OpenAI та Gemini від Google, кожен з яких має власні сильні сторони.

Сценарії застосування AI-інструментів у розробці лендингу. Універсальний характер сучасних LLM-асистентів дозволяє застосовувати їх практично на всіх етапах розробки веб-продукту. Типові сценарії, що становлять найбільший інтерес у контексті цієї роботи, включають:

- *генерацію стартової структури компонентів* — наприклад, початкового скелета Blade-шаблонів, моделей Eloquent, контролерів Laravel за наданим описом структури даних;

- *написання SQL-запитів та міграцій* — на основі опису потрібної структури таблиці або трансформації даних;
- *пояснення наявного коду* — особливо цінне при роботі з незнайомими бібліотеками (зокрема Orchid Platform) або при аналізі помилок;
- *рефакторинг та оптимізація* — пропозиції щодо покращення читабельності, винесення повторюваної логіки в окремі методи, виправлення невідповідностей стилю;
- *генерацію тестових даних* — наповнення сидерів та фікстур правдоподібним контентом;
- *написання документації та коментарів* — переклад коду на природну мову для подальшої передачі замовнику;
- *проектування дизайну та генерація HTML/CSS-розмітки* — за допомогою спеціалізованих інструментів типу Google Stitch або через генерацію коду універсальною LLM.

Обмеження AI-інструментів. Незважаючи на стрімкий прогрес, AI-інструменти на сучасному етапі мають низку суттєвих обмежень, які впливають на можливості їх практичного застосування. Ці обмеження можна об'єднати у чотири основні групи.

Обмеження якості згенерованого коду. Код, згенерований мовною моделлю, може містити логічні помилки, неоптимальні рішення, прихильність до застарілих практик або порушення прийнятих у конкретній екосистемі патернів. Особливою категорією ризиків є потенційні вразливості безпеки — від банальних, як відсутність валідації введених даних, до більш складних, як некоректне використання криптографічних примітивів. Це визначає необхідність обов'язкової верифікації згенерованого коду кваліфікованим розробником і фактично виключає сценарій «прийняти результат як є».

Контекстні обмеження. Універсальна LLM не має повного уявлення про специфіку конкретного проєкту: внутрішні домовленості

команди, особливості бізнес-логіки, неявні вимоги замовника. Це може призводити до ситуацій, коли згенерований код є формально коректним і навіть елегантним, але функціонально не відповідає очікуванням — наприклад, обчислює правильний результат за неправильною формулою, що випадково збігається з очікуваним результатом для тестового прикладу. Зменшити цей ефект можна детальним описом контексту в запиті, що, втім, переносить навантаження на розробника [15].

Залежність від якості постановки завдання. Результат генерації значною мірою визначається тим, як саме сформульовано запит. Нечіткі або занадто загальні формулювання призводять до отримання шаблонних рішень, що формально відповідають запиту, але не вирішують реальної задачі. Це зумовлює появу нової практичної компетентності — *prompt engineering* — навичок формулювання вимог до AI-інструмента, що формуються переважно емпіричним шляхом і поступово стають невіддільною частиною професійної підготовки розробника [13].

Архітектурні обмеження. Складні архітектурні рішення, інтеграція множинних компонентів, проектування великих систем і довгострокові рішення про технологічний стек залишаються завданнями, де AI може виступати лише як консультант, але не як виконавець. Сучасні LLM ефективніші на рівні окремих функцій, класів або компонентів, ніж на рівні системи в цілому, і навіть з огляду на можливість обробляти великі обсяги коду, їх рекомендації потребують критичної оцінки [7].

Інструменти, обрані для цієї роботи. З урахуванням викладеного для розробки лендингу факультету застосовувалися такі AI-інструменти:

- **Claude (Anthropic)** [16] — як основний асистент для генерації серверного коду на Laravel, написання міграцій, налаштування Orchid-екранів, оптимізації Blade-шаблонів та пояснення помилок.

Робота з Claude велася у форматі діалогових сесій з поетапним уточненням завдань.

– **Google Stitch** [20] — як експериментальний інструмент для генерації варіантів дизайну окремих секцій сторінки на основі текстових описів. Результати Stitch використовувалися як референси для подальшої ручної реалізації у Figma та Blade-шаблонах.

– **Gemini (Google)** — як допоміжний консультативний інструмент при швидкому пошуку відповідей на технічні питання та порівнянні підходів. Gemini не використовувався як основне джерело генерації коду.

Конкретний характер застосування цих інструментів, виявлені практичні особливості та узагальнені спостереження детально розглядаються у підрозділі 2.6.

Висновок до підрозділу. AI-інструменти на сучасному етапі є потужним доповненням до арсеналу веб-розробника, здатним суттєво прискорити окремі етапи створення продукту — від проєктування інтерфейсів до генерації серверного коду та документації. Водночас вони не є повноцінною заміною кваліфікованого розробника і потребують обов'язкової верифікації та доопрацювання отриманих результатів. Ключовою компетентністю розробника, що використовує AI-інструменти, стає здатність точно формулювати завдання, критично оцінювати згенерований результат та інтегрувати його з рештою кодової бази. У межах цієї роботи AI-інструменти застосовувалися як допоміжний засіб на ряді етапів розробки лендингу, що дозволило отримати практичний досвід їх використання у реальному проєкті — узагальнення цього досвіду наведено у Розділі 2.

Висновки до Розділу 1

У першому розділі кваліфікаційної роботи розглянуто теоретичні та проєктні засади розробки лендингу факультету закладу вищої освіти, а також сформовано концептуальну і технологічну базу для практичної реалізації, представленої у Розділі 2.

1. Уточнено поняття лендингу та його застосовність у контексті веб-комунікації ЗВО. Встановлено, що лендинг є типом веб-сторінки з односторінковою архітектурою, обмеженою навігацією та орієнтацією на конкретну дію відвідувача. Хоча у класичному маркетинговому розумінні лендинг оптимізується під вимірювані метрики конверсії, його структурні рішення можна обґрунтовано переносити на ресурси факультетів ЗВО — за умови врахування специфіки академічної комунікації та регуляторних обмежень освітнього сектору. Для факультетів ІТ-спрямування створення спеціалізованого лендингу набуває особливого значення, оскільки якість веб-представництва сприймається цільовою аудиторією як непрямий індикатор рівня підготовки.
2. Реконструйовано чотири групи інформаційних потреб абітурієнтів, що мають бути забезпечені факультетським лендингом: інформація про освітні програми та спеціальності; умови вступу і фінансові аспекти; навчальне середовище та інфраструктуру; канали комунікації з факультетом. Виокремлено вимоги до користувацького інтерфейсу, що впливають із поведінкових особливостей цільової аудиторії: адаптивність до пристроїв різного типу, швидкодія завантаження, структурованість викладу, помітність елементів заклику до дії та доступність ресурсу (web accessibility) як для звичайних користувачів, так і для осіб з функціональними обмеженнями. Сформовано матрицю відповідності між групами потреб та секціями лендингу, що становить основу для проєктування його структури у Розділі 2.

3. Здійснено аналіз структурних рішень лендингів трьох провідних українських IT-шкіл — Mate Academy, GoIT та Hillel IT School — як референтної вибірки освітніх продуктів, що використовують лендинг у класичному маркетинговому значенні. На основі аналізу виокремлено сім стійких структурних елементів, спільних для всіх розглянутих ресурсів: hero-секція з ціннісною пропозицією, заклик до дії, каталог освітніх продуктів, прив'язка до професійних результатів, соціальний доказ, FAQ-секція та форма зворотного зв'язку. Водночас зафіксовано принципові відмінності між комерційним та академічним сегментами, що визначає необхідність переосмислення, а не прямого копіювання референтних рішень при перенесенні їх на факультетський лендинг.
4. Обґрунтовано вибір технологічного стеку для реалізації лендингу. Стек складається з PHP 8.2 та Laravel 12 як основи серверної частини, Orchid Platform як інструмента побудови адміністративної панелі, MySQL як реляційної СУБД, Apache як HTTP-сервера, Blade як шаблонізатора, astrotomic/laravel-translatable для забезпечення двомовності, Vite для збірки фронтенд-ресурсів, та Docker з Docker Compose для контейнеризації і розгортання. Сформульовано чотири критерії вибору технологій для проєктів освітнього сектору: зрілість компонентів, доступність спільнотної підтримки, можливість автономного управління контентом замовником та переносність розгортання. Прийняті рішення обґрунтовано порівнянням з альтернативами (PostgreSQL, Filament, Nova, SPA-фреймворки) із зазначенням підстав для відхилення кожної.
5. Розглянуто сучасний стан AI-інструментів у веб-розробці та їх потенціал як допоміжного засобу. Виокремлено чотири категорії таких інструментів — універсальні LLM-асистенти, спеціалізовані

асистенти у середовищі розробки, генератори інтерфейсів та платформи повного циклу. Зафіксовано основні обмеження сучасних AI-інструментів: обмеження якості згенерованого коду, контекстні обмеження, залежність від якості постановки завдання та архітектурні обмеження. Зроблено висновок, що AI-інструменти на сучасному етапі є потужним доповненням до арсеналу веб-розробника, проте не замінюють його кваліфікацію, а зміщують характер роботи у бік формулювання завдань та критичної оцінки результату. Для розробки лендингу обрано три інструменти: Claude як основний асистент генерації коду, Google Stitch для експериментів з дизайном, Gemini як консультативний інструмент.

Сформовані у цьому розділі теоретичні засади, узагальнення референтної практики, технологічний стек та підхід до застосування AI-інструментів становлять основу для практичної розробки лендингу, що детально розглядається у Розділі 2.

РОЗДІЛ 2

РОЗРОБКА ТА РЕАЛІЗАЦІЯ ЛЕНДИНГУ ФАКУЛЬТЕТУ

2.1. Проектування лендингу: інформаційна архітектура, контент, дизайн-система

Перш ніж переходити до програмної реалізації, було виконано підготовчий етап, на якому визначено інформаційну архітектуру лендингу, сформовано контент спільно з представниками факультету та закладено основу дизайн-системи. Цей етап є логічним продовженням теоретичної частини роботи: реконструйовані у підрозділі 1.2 інформаційні потреби абітурієнтів та узагальнені у підрозділі 1.3 структурні рішення лендингів ІТ-шкіл становлять основу для рішень щодо складу та послідовності секцій сторінки.

Інформаційна архітектура. Лендинг факультету комп'ютерних наук, фізики та математики ХДУ має односторінкову (one-page) архітектуру з вертикальним послідовним розгортанням контенту, навігацією через якірні посилання у верхній панелі та модальними вікнами для деталізації окремих елементів. Сторінка складається з функціональних секцій, послідовність яких відображає природний порядок сприйняття інформації потенційним абітурієнтом: від формування первинного враження — через презентацію освітніх можливостей — до конкретних кроків щодо вступу та контактування з факультетом. Загальний склад секцій представлено у таблиці 2.1.

Таблиця 2.1

Структура секцій лендингу факультету

№	Секція	Функціональне призначення
1	Header	Логотип, навігація по секціях, контактний телефон,

№	Секція	Функціональне призначення
		перемикач мови
2	Него-секція	Ключова ціннісна пропозиція, заклик до дії, карусель зображень
3	Екосистема дистанційного навчання	Презентація інфраструктури онлайн-навчання факультету
4	Переваги факультету	Опис ключових переваг навчання, орієнтований на абітурієнта
5	Освітні програми	Каталог семи бакалаврських та п'яти магістерських програм з фільтрацією за рівнем
6	Кар'єрні перспективи	Партнери, відгуки випускників, статистичні показники
7	Про факультет	Історична довідка та система цінностей факультету
8	Міжнародна діяльність	Інформація про дослідницькі проекти факультету
9	Міжнародні можливості	Англомовні програми та умови навчання для іноземних студентів
10	Викладачі	Представлення науково-педагогічного складу за кафедрами
11	Для абітурієнтів	Покрокові інструкції щодо вступу на бакалаврат та магістратуру
12	FAQ	Відповіді на типові запитання, згруповані за категоріями
13	Контактна форма та футер	Форма зворотного зв'язку, контактні дані, посилання на соцмережі

Структура побудована з урахуванням матриці відповідності інформаційних потреб і секцій лендингу, представленої у підрозділі 1.2 (таблиця 1.1). Кожна з виокремлених у Розділі 1 груп інформаційних потреб абітурієнта забезпечена щонайменше однією секцією лендингу, а в окремих випадках — комбінацією декількох секцій, що дозволяє подати інформацію з різних кутів (наприклад, формат навчання представлений як у секції «Екосистема дистанційного навчання», так і у секції «Переваги факультету» та у відповідних категоріях FAQ).

Особливості ключових секцій. Окремої уваги при проектуванні потребували секції, що мають нетривіальну функціональну логіку.

Hero-секція є першим візуальним елементом, з яким взаємодіє відвідувач, і фактично визначає рішення про подальше вивчення сторінки. Її структура включає ключовий заголовок із ціннісною пропозицією факультету, підзаголовок із деталізацією, два заклики до дії, диференційовані за категорією аудиторії — для українських вступників та для іноземних студентів, — а також карусель тематичних зображень, реалізовану через бібліотеку Embla Carousel. Контент hero-секції є динамічним і керується через окремий екран адміністративної панелі «Слайди».

Секція «Освітні програми» є найбільш функціонально складною частиною лендингу. Вона представляє сім бакалаврських та п'ять магістерських освітніх програм факультету у межах шести спеціальностей, з можливістю фільтрації за рівнем вищої освіти. Для кожної програми відображається назва спеціальності, тривалість навчання, форма навчання, орієнтовна вартість контрактного навчання, кваліфікація, перелік ключових обов'язкових дисциплін та орієнтовні професійні перспективи випускника. Навігація між програмами реалізована через табовий перемикач рівнів освіти та бічне меню зі списком програм поточного рівня на десктопній версії; на мобільній версії бічне меню замінюється кастомним випадаючим списком (select) для збереження компактності розмітки.

Секція FAQ реалізована з використанням системи категорій (Вступ, Документи, Онлайн-навчання, Вартість, Іноземцям), що дозволяє абітурієнтам швидко знаходити інформацію, релевантну їхній ситуації. Кожна категорія містить акордеон з парами «запитання — відповідь» та окремий блок із прикріпленими відеоматеріалами, що візуалізують відповіді на найпопулярніші запитання.

Секція «Для абітурієнтів» реалізована у форматі покрокової інструкції, з диференціацією за освітнім рівнем — окремі потоки для бакалаврського (НМТ/ЗНО, подача документів, зарахування) та магістерського (ЄВІ/ЄФВВ, подача документів, зарахування) вступу. Такий формат відповідає очікуванням абітурієнтів отримати структуровану практичну інформацію, а не виписки з нормативних документів.

Адаптивність та мобільна версія. Лендинг реалізовано за принципом *mobile-first* з прогресивним поліпшенням для більших екранів [11]. Контрольні точки CSS-медіазапитів встановлено для трьох категорій пристроїв: мобільні (до 768px), планшетні (768–1024px) та десктопні (від 1024px). На мобільній версії передбачено такі ключові адаптації: основна навігація замінюється на «бургер-меню» з виїздом збоку; бічне меню секції освітніх програм трансформується у випадаючий список; багатостолбкові блоки переходять в одностолбкову структуру; інтерактивні елементи (кнопки, лінки в навігації) збільшуються до розміру, що забезпечує зручну взаємодію через дотик.

Двомовність. Лендинг функціонує у двох мовних версіях — українській (основна) та англійській. Розмежування мовних версій реалізовано на рівні URL: основні роуті доступні за кореневим шляхом (/), англійські — за префіксом /en/. Перемикач мови у шапці змінює базовий URL поточної сторінки, зберігаючи якірне посилання на поточну секцію. На рівні бази даних двомовність забезпечує пакет *astrotoomic/laravel-translatable* [17]: текстовий контент розділено на мовнонезалежну частину (зображення, URL, числові значення) у основних таблицях та перекладну частину (заголовки, описи, тексти) у таблиці *translations*. Це дозволяє додавати нові мови у майбутньому без зміни структури основних таблиць.

Контент. Інформаційне наповнення лендингу формувалося спільно з представниками факультету як замовником і охоплює:

інформацію про дванадцять освітніх програм (повна назва, кваліфікація, тривалість, форма навчання, вартість, обов'язкові дисципліни, кар'єрні перспективи); історичну довідку про факультет; систему цінностей; інформацію про міжнародні дослідницькі проекти факультету; відомості про науково-педагогічний склад кафедр; типові запитання з відповідями за п'ятьма категоріями; контактні дані приймальної комісії та факультету. Окремо слід зазначити, що частина контенту на момент здачі цієї роботи перебуває у процесі поетапної заміни шаблонних матеріалів реальними даними — насамперед це стосується розділів «Кар'єрні перспективи» (відгуки випускників, історії успіху, логотипи компаній-партнерів) та текстових описів окремих освітніх програм. Технічна реалізація передбачає, що такі оновлення виконуються представниками факультету самостійно через адміністративну панель, без залучення розробника, що відповідає вимозі автономності управління контентом, сформульованій у підрозділі 1.4.

Дизайн-система. На етапі проектування у графічному редакторі Figma було визначено базові елементи дизайн-системи лендингу. Кольорова палітра ґрунтується на фірмових кольорах ХДУ: основний колір #4E008E (фірмовий фіолетовий), додатковий #16325C (темно-синій), акцентний #005495 (для СТА та посилань). Базова типографіка побудована на одному шрифтовому сімействі для забезпечення візуальної єдності, з ієрархією розмірів для заголовків Н1–Н3 та основного тексту. Сітка макета — стандартна 12-колонкова з контейнером максимальної ширини 1280px на десктопі та повноекранним відображенням на мобільних. Базові UI-компоненти (кнопки, поля введення, випадаючі списки, картки, акордеони) визначено у Figma як перевикористовувані компоненти, що забезпечило візуальну узгодженість на етапі верстки.

Висновок до підрозділу. Спроектowana інформаційна архітектура поєднує тринадцять функціональних секцій, послідовність яких

відповідає природному порядку сприйняття інформації абітурієнтом, із трьома основними патернами візуального представлення складних даних (табовий перемикач, акордеон, покрокова інструкція). Дизайн-система ґрунтується на фірмових кольорах ХДУ і спроектована з урахуванням адаптивності, доступності та можливості подальшого масштабування. Усі змістові та візуальні елементи (тексти, зображення, посилання, конфігурації) реалізовано як динамічні сутності, керовані через адміністративну панель. Технічна реалізація описаної структури розглядається у подальших підрозділах: фронтенд — у 2.2, серверна частина — у 2.3, адміністративна панель — у 2.4.

2.2. Реалізація фронтенду: верстка, адаптивність, інтерактивні елементи

Клієнтська частина лендингу реалізована з використанням стеку HTML5, CSS3 та JavaScript [6; 9] у поєднанні з шаблонізатором Blade, інтегрованим у Laravel. Збірка та оптимізація фронтенд-ресурсів виконується через Vite [26]. У цьому підрозділі описано організацію верстки, реалізацію адаптивності та інтерактивних елементів сторінки, а також підхід до забезпечення семантичності розмітки відповідно до вимог web accessibility, сформульованих у підрозділах 1.2 та 1.4.

Структура фронтенд-частини проєкту. Фронтенд організовано за компонентним принципом: кожна секція лендингу реалізована як окремий Blade-шаблон у каталозі `resources/views/components/`, що підключається до головного шаблону сторінки через директиви Blade. Стилi CSS згруповано за модулями — окремий файл стилів на кожну значущу секцію або UI-компонент, з імпортом усіх модулів у головний CSS-файл. Аналогічний принцип застосовано для JavaScript: окремий модуль на кожен інтерактивний компонент, що дозволяє

підтримувати кодову базу без розростання монолітних файлів. Зведена структура каталогів фронтенду наведена в таблиці 2.2.

Таблиця 2.2

Структура каталогів фронтенд-частини

Каталог	Призначення
resources/views/landing.blade.php	Головний шаблон лендингу, що включає всі секції
resources/views/components/	Blade-шаблони окремих секцій (header, hero, programs, faq, footer тощо)
resources/css/	CSS-модулі по одному на секцію + base.css для глобальних стилів
resources/js/	JavaScript-модулі для інтерактивних компонентів
resources/lang/	Файли локалізації (uk, en)
public/build/	Скомпільовані продакшн-ресурси, що генеруються Vite

Принципи верстки. При реалізації верстки лендингу дотримувались таких принципів:

- *семантична розмітка HTML5*: використання тегів <header>, <nav>, <main>, <section>, <article>, <footer> для передачі структури сторінки програмам екранного читання та пошуковим роботам; послідовна ієрархія заголовків від <h1> (єдиний на сторінці, у hero-секції) до <h2> (заголовки секцій) і <h3> (підзаголовки в межах секцій);
- *уникнення зайвої вкладеності*: пласка структура DOM-дерева без надлишкових обгорткових <div>, що полегшує читання коду та підвищує продуктивність рендерингу;
- *клас-орієнтоване стилювання*: усі стилі прив'язано до класів, без використання інлайнових стилів або стилізації за тегами елементів;

цей підхід забезпечує передбачуваність каскаду та полегшує подальшу підтримку;

- *модульна організація стилів*: класи групуються за функціональними секціями та компонентами, що забезпечує локалізацію змін у межах одного модуля та запобігає колізіям імен між секціями.

Стили CSS. Для написання стилів використано чистий CSS3 без препроцесорів (Sass, Less) та утиліт-фреймворків (Tailwind). Це рішення зумовлене двома міркуваннями. По-перше, обсяг стилів лендингу не настільки великий, щоб переваги препроцесорів (змінні, міксини, вкладеність) виправдовували додаткову складність збірки — сучасний CSS3 нативно підтримує змінні (CSS custom properties), вкладеність (CSS Nesting) та функцію `calc()`, що покриває основні потреби. По-друге, виключення фреймворків-утиліт дозволило отримати компактну стилеву базу, орієнтовану саме під потреби цього проєкту, без зайвого «мертвого коду» з невикористаних класів. Кольорова палітра, типографіка та базові параметри сітки винесені у CSS-змінні в `:root`, що дозволяє централізовано керувати дизайн-токенами.

Адаптивність. Адаптивність реалізована через CSS-медіазапити з трьома основними контрольними точками: мобільні пристрої (до 768px), планшети (768–1024px) та десктоп (від 1024px). Базові стилі написано для мобільної версії (підхід *mobile-first* [11]), додаткові стилі для планшетів і десктопу підключаються через `@media (min-width: ...)`. Ключові адаптивні рішення включають: трансформацію основної навігації у «бургер-меню» з виїздом збоку на мобільній версії; заміну бічного меню секції освітніх програм на випадаючий список на вузьких екранах; перехід багатоколонкових блоків (як «Топ-5 переваг», система цінностей факультету) в одноколонкову структуру; збільшення розмірів кнопок та інтерактивних елементів для зручності дотикової взаємодії.

Інтерактивні елементи. Сторінка містить низку інтерактивних компонентів, реалізованих засобами JavaScript. Огляд ключових компонентів наведено в таблиці 2.3.

Таблиця 2.3

Інтерактивні компоненти фронкенду

Компонент	Призначення	Реалізація
Карусель hero-секції	Послідовний показ слайдів з можливістю ручної навігації	Бібліотека Embla Carousel
Перемикач рівнів освіти (таби)	Перемикання між бакалаврським та магістерським рівнями	Власна реалізація на класах CSS та обробниках кліків
Бічне меню програм	Вибір конкретної освітньої програми на десктопі	Власна реалізація
Випадаючий список програм	Вибір програми на мобільній версії	Кастомний select з власною стилізацією
Таби FAQ	Перемикання між категоріями	Власна реалізація на класах
Акордеон FAQ	Розкриття/згортання відповідей на запитання	Власна реалізація з CSS-анімацією
Мобільне меню (бургер)	Основна навігація на мобільній версії	Власна реалізація
Контактна форма	Відправлення заявки від абітурієнта	Fetch-запит на серверну частину з валідацією
Перемикач мови	Зміна локалі сторінки (uk/en)	Перенаправлення на відповідний URL з префіксом
Модальне вікно успішної відправки	Підтвердження успішного приймання заявки	Завантаження HTML-фрагмента через Fetch API

З усіх перелічених компонентів зовнішня бібліотека використана лише для каруселі — Embla Carousel обрано як легковісний, добре документований інструмент без зайвих залежностей. Решта інтерактивних елементів реалізована на чистому JavaScript без

додаткових бібліотек: складність кожного окремого компонента не виправдовує підключення фреймворку, а сумарна кодова база залишається компактною.

Web accessibility у реалізації. Дотримання принципів доступності реалізовано на кількох рівнях відповідно до вимог стандарту WCAG 2.1 [27]. На рівні розмітки використано семантичні теги HTML5 та послідовну ієрархію заголовків. Для всіх змістовних зображень задано атрибут `alt` з осмисленим описом; декоративні зображення мають порожній `alt=""` для коректної обробки програмами екранного читання. Для інтерактивних елементів, що не мають прямих семантичних аналогів у HTML (карусель, акордеон, кастомний випадаючий список), застосовано ARIA-атрибути для передачі стану та ролі: `aria-expanded` для акордеону, `aria-label` для кнопок-іконок, `role="tablist"` та `role="tab"` для табових перемикачів. Клавіатурна навігація забезпечена для всіх інтерактивних елементів — фокус послідовно проходить через елементи сторінки у логічному порядку, з видимим індикатором поточного фокусу. Кольоровий контраст тексту та фону відповідає вимогам WCAG 2.1 рівня AA (співвідношення не менше 4.5:1 для основного тексту).

Збірка та оптимізація. Збірка фронтенд-ресурсів виконується через Vite з конфігурацією, заданою у `vite.config.js`. У режимі розробки Vite запускає dev-сервер з підтримкою `hot module replacement`, що дозволяє бачити зміни у браузері без перезавантаження сторінки. У продакшн-режимі Vite виконує мініфікацію CSS і JavaScript, об'єднання модулів у бандли, генерацію файлів з хешем у назві (для коректного управління кешуванням браузера) та оптимізацію шляхів імпорту. Підключення згенерованих ресурсів до Blade-шаблонів виконується через директиву `@vite()`, що автоматично підставляє актуальні шляхи з маніфесту збірки.

Висновок до підрозділу. Фронтенд лендингу реалізовано на чистому стеку HTML5 + CSS3 + JavaScript із серверним рендерингом через Blade, без використання SPA-фреймворків або CSS-фреймворків-утиліт. Розмітка дотримується принципів семантичності та доступності відповідно до вимог WCAG; адаптивність забезпечена через мобільний підхід з трьома контрольними точками; інтерактивні елементи реалізовано переважно власними засобами з мінімальним залученням зовнішніх бібліотек. Збірка ресурсів виконується через Vite, що забезпечує сучасний workflow розробки та оптимізовану продакшн-збірку.

2.3. Реалізація серверної частини на Laravel

Серверна частина лендингу реалізована на фреймворку Laravel 12 [22] з використанням архітектурного патерну MVC (Model-View-Controller) [14]. У цьому підрозділі описано структуру серверної частини: модель даних, систему маршрутизації, контролери, обробку контактної форми та реалізацію двомовності.

Структура бази даних. Схема бази даних спроектована з урахуванням двох вимог: забезпечити динамічне керування всім контентом лендингу через адміністративну панель та підтримувати двомовність контенту через пакет `astrotoomic/laravel-translatable` [17]. Усі таблиці створено через систему міграцій Laravel, що забезпечує версіонування структури бази даних та можливість її відтворення на будь-якому середовищі однією командою `php artisan migrate`. Огляд ключових таблиць наведено в таблиці 2.4.

Таблиця 2.4

Ключові таблиці бази даних лендингу

Таблиця	Призначення
---------	-------------

Таблиця	Призначення
educational_programs	Освітні програми факультету (мовнонезалежні поля: код, рівень, форма навчання, вартість, статус активності, порядок сортування)
sliders, slides	Каруселі hero-секції та слайди у складі каруселей
faq_categories, faq_items, faq_materials	Категорії FAQ, окремі питання-відповіді та прикріплені до них корисні матеріали
leads	Заявки абітурієнтів, прийняті через контактну форму
configs	Загальні налаштування сайту
seo_metrics	Користувацькі скрипти для інжекції у <head>
translations	Перекладні текстові поля для всіх моделей
users, roles, role_users	Користувачі адмін-панелі та система ролей (Orchid Platform)
attachments, attachmentable	Файлові вкладення з поліморфними зв'язками (Orchid)
sessions, cache, jobs, notifications тощо	Технічні таблиці Laravel для роботи сесій, кешу, черг та системи сповіщень

Кожна сутність, що містить перекладні текстові поля (назва, опис, кваліфікація тощо), реалізована як пара «основна таблиця + записи у таблиці translations». Основна таблиця містить мовнонезалежні дані (числа, посилання, файли, статуси, порядок сортування), таблиця translations — текстові поля у форматі «ідентифікатор сутності + код мови + ім'я поля + значення». Такий підхід відрізняється від поширеного, але менш масштабованого варіанту з додаванням дублювальних колонок для кожної мови (наприклад, name_uk, name_en) і дозволяє додавати нові мови без зміни структури основних таблиць.

Модель даних та Eloquent. Робота з базою даних виконується через ORM Eloquent — стандартний інструмент Laravel, що відображає таблиці бази даних на PHP-класи. Для кожної основної сутності

лендингу створено окремий клас Eloquent-моделі, розміщений у каталозі `app/Models/`. Перекладні моделі реалізують патерн через `trait Translatable` пакета `astrotomic`. Кожна така модель оголошує властивість `$translatedAttributes` — масив імен полів, що зберігаються у таблиці `translations`. При зверненні до перекладного поля Eloquent автоматично повертає значення для поточної локалі або для локалі за замовчуванням, якщо переклад для поточної відсутній. Це дозволяє в шаблонах писати простий код виду `$program->name` замість явного звернення до конкретної локалі.

Маршрутизація. Конфігурація маршрутизації знаходиться у файлі `routes/web.php` і реалізує двомовну схему через групи маршрутів з префіксом локалі. Структура файлу організована таким чином: набір клієнтських роутів виноситься в окрему замикаючу функцію (`$landingRoutes`), що далі підключається у двох контекстах — без префікса (українська локаль за замовчуванням) та з префіксом `/en/` (англійська локаль). У обох випадках застосовується `middleware SetLocale`, який встановлює поточну локаль Laravel перед обробкою запиту. Зведення основних роутів наведено в таблиці 2.5.

Таблиця 2.5

Основні маршрути клієнтської частини

Метод	URL	Контролер та дія	Призначення
GET	/	LandingController@index	Головна сторінка, українська версія
GET	/en	LandingController@index	Головна сторінка, англійська версія
POST	/lead	LeadController@store	Приймання заявки з контактної форми
POST	/en/lead	LeadController@store	Те саме для англійської версії
POST	/modal/success	LandingController	Завантаження

Метод	URL	Контролер та дія	Призначення
		@getSuccessModal	модального вікна успіху
POST	/en/modal/success	LandingController @getSuccessModal	Те саме для англomовної версії

Адміністративні маршрути зареєстровані окремо у файлі `routes/platform.php`, який є стандартним місцем розміщення роутів для пакета Orchid Platform. Детально структура адміністративних роутів розглядається у підрозділі 2.4.

Контролери. Серверна частина включає два основні контролери для клієнтської частини, розміщені у каталозі `app/Http/Controllers/`. `LandingController` відповідає за обробку запитів до головної сторінки: завантажує з бази даних усі необхідні сутності (освітні програми, FAQ, слайди, налаштування, переклади інтерфейсу), формує єдиний масив даних і передає його у Blade-шаблон головної сторінки. Окремий метод `getSuccessModal` обробляє запити, що виконуються з клієнта через Fetch API, і повертає HTML-фрагмент модального вікна, яке відображається після успішного приймання заявки. `LeadController` обробляє POST-запити з контактної форми: виконує валідацію вхідних даних, створює запис у таблиці `leads` та повертає JSON-відповідь зі статусом виконання.

Обробка контактної форми. Контактна форма є основною точкою конверсії лендингу: абітурієнти залишають свої контактні дані для подальшої консультації з представниками приймальної комісії. Реалізація включає клієнтську валідацію засобами JavaScript (для миттєвого зворотного зв'язку при заповненні полів) та обов'язкову серверну валідацію через стандартний механізм `Laravel Validation` у контролері. Серверна валідація перевіряє: обов'язковість полів імені та електронної пошти; коректність формату електронної пошти; наявність

хоча б одного з контактних каналів (Telegram або номер у Viber); коректність обраного освітнього рівня. У разі успішної валідації запис створюється у таблиці `leads`, з якої адміністратори факультету переглядають заявки через відповідний екран Orchid. У разі помилки повертається JSON-відповідь з переліком полів, що не пройшли валідацію, та відповідними повідомленнями локалізованими мовою поточної версії сторінки.

Локалізація інтерфейсу. Окрім перекладного контенту, що зберігається в базі даних, серверна частина містить локалізаційні рядки для статичних елементів інтерфейсу — кнопок, повідомлень валідації, написів на формах. Ці рядки розміщені у файлах локалізації `lang/uk/common.php` та `lang/en/common.php`, що повертають PHP-масиви ключ-значення для кожної мови. Звернення до перекладів у Blade-шаблонах виконується через функцію `__( 'common.key' )`, що повертає значення для поточної локалі. Перемикання локалі виконується middleware `SetLocale` на рівні маршрутів, що забезпечує коректне відображення інтерфейсу відповідно до обраної мови.

Безпека та валідація. При реалізації серверної частини дотримано стандартних практик безпеки Laravel [14]. CSRF-захист увімкнено за замовчуванням для всіх POST-запитів — токен автоматично додається у форми через директиву Blade `@csrf` і перевіряється middleware `VerifyCsrfToken`. Захист від SQL-ін'єкцій забезпечує Eloquent через параметризовані запити — у коді не використовуються «сирі» SQL-запити з конкатенацією користувацьких даних. Захист від XSS забезпечує Blade за замовчуванням — усі дані, що виводяться через подвійні фігурні дужки `{{ $value }}`, автоматично екрануються; неекрановане виведення `{!! $value !!}` використовується лише там, де воно справді необхідне (зокрема для відображення HTML, відредагованого через редактор у адмінпанелі), та з попереднім контролем джерела даних.

Для роботи з даними використовується СУБД MySQL [25], розгорнута у окремому Docker-контейнері, технічні аспекти якого розглядаються у підрозділі 2.5.

Висновок до підрозділу. Серверна частина лендингу реалізована на Laravel 12 за класичною архітектурою MVC з ORM Eloquent для роботи з базою даних. Двомовність забезпечена на двох рівнях: на рівні даних — через пакет `astrotomic/laravel-translatable` з винесенням перекладних полів в окрему таблицю; на рівні маршрутизації — через префікс локалі в URL з обробкою middleware `SetLocale`. Обробка контактної форми як основної точки конверсії включає клієнтську і серверну валідацію та збереження заявок у базі для подальшої роботи приймальної комісії. Стандартні практики безпеки Laravel (CSRF, екранування Blade, параметризовані запити Eloquent) дозволили реалізувати захист від поширених вебуразливостей без додаткових налаштувань. Робота з адміністративною панеллю, реалізованою через Orchid Platform, детально розглядається у наступному підрозділі.

2.4. Адміністративна панель на Laravel Orchid

Адміністративна панель лендингу побудована на пакеті Orchid Platform [24] — спеціалізованому інструменті для створення адміністративних інтерфейсів поверх Laravel, вибір якого обґрунтовано у підрозділі 1.4. У цьому підрозділі описано архітектуру адмін-панелі, склад її екранів, реалізацію CRUD-операцій для основних сутностей лендингу та практичні особливості роботи з Orchid 14, виявлені у процесі розробки.

Архітектура Orchid та концепція Screen. На відміну від класичного підходу Laravel, де адмін-панель будується як набір контролерів, шаблонів і форм, Orchid пропонує декларативну архітектуру на основі компонентів типу Screen. Screen — це PHP-клас,

що описує адміністративну сторінку як набір методів: метод `query()` повертає дані для відображення, метод `layout()` визначає UI-компоненти (таблиці, форми, фільтри, кнопки), метод `commandBar()` задає набір дій верхньої панелі. Orchid самостійно генерує відповідний HTML, обробляє користувацький введення, виконує валідацію та зв'язує форми з моделями Eloquent. Розробник зосереджується на описі того, **що** має бути на сторінці, а не на тому, **як** це відмалювати. Класи Screen розміщуються у каталозі `app/Orchid/Screens/` з тематичною підкаталогізацією (Programs, Faq, Slider, Role, User, SEO, Config).

Маршрутизація адмін-панелі. Усі адміністративні маршрути зареєстровано у файлі `routes/platform.php`, що є стандартним розміщенням роутів для Orchid Platform. Маршрути базуються на методі `Route::screen()`, який реєструє Screen-клас як обробник конкретного URL. Кожен маршрут має іменну ідентифікацію (`name`) для подальшого використання у шаблонах та коді, а також ланцюжок крихт навігації (`breadcrumbs`) для відображення шляху на сторінці. Для більшості ресурсів реалізовано три типові маршрути: список сутностей (`/ {resource}`), форма створення (`/ {resource} /create`) та форма редагування (`/ {resource} / {id} /edit`). Огляд функціональних екранів адмін-панелі наведено у таблиці 2.6.

Таблиця 2.6

Функціональні екрани адміністративної панелі

Розділ	Екрани	Призначення
Слайдери	SliderListScreen, SliderEditScreen, SlideEditScreen	Керування каруселями hero-секції та окремими слайдами
Освітні програми	EducationalProgramListScreen, EducationalProgramEditScreen	CRUD освітніх програм з підтримкою двомовного контенту
FAQ	FaqCategoryListScreen,	Керування категоріями,

Розділ	Екрани	Призначення
	FaqCategoryEditScreen, FaqItemEditScreen, FaqMaterialEditScreen	питаннями та матеріалами FAQ
Заявки	LeadListScreen	Перегляд заявок від абітурієнтів
Переклади	TranslationListScreen	Керування локалізаційними рядками інтерфейсу
SEO Метрики	SeoMetricListScreen	Користувачькі скрипти для інжекції у <head>
Налаштування сайту	ConfigEditScreen	Глобальні налаштування лендингу
Користувачі та ролі	UserListScreen, UserEditScreen, RoleListScreen, RoleEditScreen	Системні екрани Orchid

Типові CRUD-операції в Orchid. Розглянемо принцип реалізації CRUD-операцій на прикладі екрана освітніх програм як найбільш складної сутності лендингу. Список програм відображається через компонент `Layout::table` з колонками: код, назва (поточною локаллю), рівень освіти, форма навчання, вартість, статус активності та контрол перетягування для зміни порядку сортування. Натискання на запис веде до екрана редагування (`EducationalProgramEditScreen`), де форма організована з використанням компонента `Layout::tabs` з двома вкладками — українська та англійська версії текстових полів. Мовнонезалежні поля (код, рівень, форма навчання, вартість, статус, порядок) винесені в окрему секцію форми над вкладками, оскільки не залежать від мови контенту. Кнопки командної панелі екрана включають типові дії: «Зберегти», «Видалити», «Повернутися до списку». Подібну організацію застосовано для категорій та елементів FAQ, слайдерів та слайдів, інших сутностей з перекладним контентом.

Особливості реалізації CMS-функціональності.

Адміністративна панель забезпечує повний контроль над контентом лендингу, виконуючи фактично функцію спеціалізованої CMS під конкретний проєкт. Ключові реалізовані можливості:

- *керування каруселями hero-секції*: створення нових каруселей, додавання та переупорядкування слайдів у їх складі, прив'язка слайдів до конкретних каруселей через адмін-екран `SliderEditScreen`;
- *керування освітніми програмами*: повний CRUD з підтримкою двомовності, drag-and-drop сортування для зміни порядку відображення програм на сторінці, активація/деактивація програм без їх видалення;
- *керування FAQ*: ієрархічна структура «категорія → питання → прикріплені матеріали», що відповідає секції FAQ на лендингу з її табовим перемикачем категорій;
- *керування заявками*: перегляд надходжень з контактної форми у форматі таблиці з можливістю фільтрації та сортування;
- *керування налаштуваннями сайту*: єдиний екран `ConfigEditScreen` для редагування глобальних параметрів — контактних телефонів, email, посилань на соцмережі, текстів сталих елементів сторінки;
- *керування SEO-скриптами*: окремий розділ для додавання довільних `<script>` та `<meta>`-тегів, що інжектуються у `<head>` лендингу — типовий випадок використання включає підключення Google Analytics, Meta Pixel, Hotjar та подібних аналітичних сервісів без необхідності зміни коду застосунку;
- *керування перекладами інтерфейсу*: окремий екран для редагування статичних локалізаційних рядків (написів на кнопках, повідомлень валідації) безпосередньо через адмін-панель, без

необхідності змінювати файли `lang/uk/common.php` та `lang/en/common.php` у файловій системі.

Авторизація та ролі. Доступ до адміністративної панелі захищений системою аутентифікації Laravel у поєднанні з системою ролей Orchid. Користувачі зберігаються у таблиці `users`, ролі — у таблиці `roles`, зв'язок «багато до багатьох» — у `role_users`. Orchid надає готовий інтерфейс для керування користувачами і ролями (`UserListScreen`, `RoleListScreen` та відповідні екрани редагування), а також механізм гранулярних дозволів (`permissions`), що дозволяють обмежити доступ до окремих екранів і дій для різних ролей. На поточному етапі експлуатації лендингу впроваджено єдину роль адміністратора з повним доступом до всіх розділів адмін-панелі. Архітектура Orchid передбачає можливість додавання нових ролей з обмеженим набором дозволів без зміни коду застосунку, що залишає простір для майбутнього розширення моделі доступу — наприклад, для введення окремих ролей редактора контенту чи модератора заявок.

Прикріплення файлів. Більшість сутностей лендингу пов'язана з графічним контентом — слайди мають фонові зображення, освітні програми та матеріали FAQ можуть містити ілюстрації. Для роботи з файлами використовується вбудована система Orchid Attachments: окремі таблиці `attachments` (метадані файлів) та `attachmentable` (поліморфні зв'язки файлів із сутностями). У формах екранів файлове поле підключається через компонент `Upload`, що забезпечує завантаження файлу, його збереження у файловому сховищі (`storage/app/public/`), створення мініатюр для зображень та формування поліморфного зв'язку з відповідною моделлю. У клієнтській частині звернення до прикріпленого файлу здійснюється через відношення Eloquent: `$slide->attachment()->url` повертає публічний URL до завантаженого зображення.

Висновок до підрозділу. Адміністративна панель лендингу побудована на пакеті Orchid Platform і реалізує дванадцять функціональних екранів, що покривають усі сутності лендингу. Декларативний підхід Orchid через концепцію Screen дозволив побудувати повнофункціональну CMS без написання клієнтського коду для адміністративного інтерфейсу — кожен екран описується одним PHP-класом, а Orchid самостійно генерує відповідний HTML, валідацію та обробку запитів. Реалізована функціональність забезпечує автономне керування контентом представниками факультету без залучення розробника, що відповідає одній з ключових вимог до проєкту, сформульованих у підрозділі 1.4. Практичні особливості роботи з Orchid 14, у тому числі складнощі, виявлені при генерації коду адмін-екранів засобами AI-інструментів, детально розглянуто у підрозділі 2.6.

2.5. Контейнеризація та розгортання

Завершальним етапом розробки лендингу є його контейнеризація та розгортання у середовищі експлуатації. У цьому підрозділі описано конфігурацію Docker [18], структуру образу застосунку, процес розгортання, а також стан продакшн-середовища, у якому лендинг наразі доступний для відвідувачів.

Обґрунтування контейнеризації. Розгортання веб-застосунків без контейнеризації передбачає ручне встановлення на сервері всіх необхідних компонентів — інтерпретатора PHP потрібної версії з певним набором розширень, веб-сервера, СУБД, інструменту керування залежностями Composer. Це створює низку проблем: невідповідність версій між середовищем розробки та продакшн-сервером, складність переносу проєкту на новий хостинг, необхідність повторного налаштування при оновленні системи. Контейнеризація розв'язує ці проблеми, упаковуючи застосунок разом з усіма його залежностями в ізольовані образи, що однаково запускаються на будь-якій машині з

установленим Docker, незалежно від хост-операційної системи та її конфігурації.

Конфігурація Docker Compose. Інфраструктура лендингу описана у файлі `docker-compose.yml`, що визначає два сервіси — `app` та `db` — з мережевими, ресурсними та залежнісними параметрами. Огляд параметрів подано у таблиці 2.7.

Таблиця 2.7

Конфігурація сервісів у `docker-compose.yml`

Параметр	Сервіс `app`	Сервіс `db`
Назва контейнера	<code>ksu-fmf-app</code>	<code>ksu-fmf-db</code>
Базовий образ	Кастомний, на основі <code>Dockerfile</code>	<code>mysql:8.0</code> (офіційний)
Внутрішній порт	80 (Apache)	3306 (MySQL)
Порт на хості	8000	3307
Том	<code>./var/www/html</code> (bind-mount)	<code>ksu_fmf_db_data:/var/lib/mysql</code>
Залежності	<code>depends_on: db</code>	—
Перезапуск	За замовчуванням	<code>restart: always</code>

Сервіс `app` побудовано з кастомного `Dockerfile`, що поетапно описує процес створення образу застосунку. За базовий обрано офіційний образ `php:8.2-apache`, який поєднує інтерпретатор PHP 8.2 та веб-сервер Apache в одному контейнері. Поверх базового образу встановлюються системні бібліотеки, необхідні для роботи PHP-розширень (`libpng-dev`, `libonig-dev`, `libxml2-dev`), та утиліти `zip`, `unzip`, `git`, `curl`. Далі виконується встановлення PHP-розширень, необхідних для роботи Laravel: `pdo_mysql` (зв'язок із MySQL [25] через PDO), `mbstring` (робота з UTF-8 рядками), `exif`

(читання метаданих зображень), `pcntl` (керування процесами для черг), `bcmath` (точна арифметика) та `gd` (обробка зображень для прев'ю в Orchid). Утиліту керування залежностями `Composer` додано до образу через копіювання її бінарного файлу з офіційного `composer`-образу (`COPY --from=composer:latest`), що є рекомендованим патерном для уникнення дублювання етапів установки.

Окремою важливою конфігураційною деталлю є зміна `DocumentRoot` веб-сервера `Apache` з типового `/var/www/html` на `/var/www/html/public`. Це здійснюється через автоматичну заміну у конфігураційних файлах `Apache` (`sites-available/*.conf` та `apache2.conf`) на етапі побудови образу. Така зміна є обов'язковою для `Laravel`-застосунків [14]: каталог `public/` містить єдину публічно доступну точку входу — файл `index.php`, який завантажує фреймворк і обробляє запит, тоді як решта коду застосунку (конфігурації, моделі, контролери, файли середовища `.env` з конфіденційними даними) має залишатися поза доступом з зовнішнього вебу. Також активується модуль `Apache mod_rewrite` (`a2enmod rewrite`), що необхідно для роботи системи маршрутизації `Laravel` — без нього всі URL мали б містити прямий шлях до `index.php`.

Для збірки фронтенд-ресурсів через `Vite` у образ додатково встановлюється `Node.js 18` з офіційного репозиторію `NodeSource`. На завершальному етапі побудови образу код проєкту копіюється у каталог `/var/www/html`, що стає робочим каталогом контейнера, а каталоги `storage/` та `bootstrap/cache/` отримують права власника `www-data:www-data` — користувача, від імені якого працює `Apache`. Без цих прав фреймворк не зміг би створювати лог-файли, кешовані шаблони `Blade` та інші службові файли під час роботи.

Сервіс `db` використовує офіційний образ `mysql:8.0` без додаткової кастомізації. Параметри підключення (назва бази,

користувач, паролі) передаються через змінні середовища з файлу `.env` проєкту, що дозволяє зберігати конфіденційні дані поза репозиторієм. Дані бази зберігаються у іменованому Docker-томі `ksu_fmf_db_data`, що забезпечує їх збереження між перезапусками контейнера. Параметр `restart: always` забезпечує автоматичний рестарт контейнера БД у разі його падіння.

Процес розгортання. Для розгортання проєкту з нуля на новому сервері потрібно виконати таку послідовність дій:

1. **Підготовка середовища.** Встановлення Docker та Docker Compose на цільовому сервері. Це єдина системна залежність — жодне інше програмне забезпечення (PHP, MySQL, Apache, Composer) не потребує окремої установки.
2. **Отримання коду.** Клонування репозиторію проєкту з системи контролю версій (`git clone`) у робочий каталог.
3. **Налаштування змінних середовища.** Копіювання файлу-шаблону `.env.example` у `.env` та редагування ключових параметрів: назва бази, ім'я користувача, пароль, корпоративний пароль `root`, ключ додатку (`APP_KEY`), посилання на застосунок (`APP_URL`).
4. **Запуск контейнерів.** Виконання команди `docker compose up -d --build`, яка автоматично будує образ застосунку з `Dockerfile`, завантажує образ MySQL з Docker Hub, створює мережу між контейнерами та запускає обидва сервіси у фоновому режимі.
5. **Налаштування Laravel.** Виконання у контейнері застосунку команд встановлення залежностей (`composer install`), генерації ключа додатку (`php artisan key:generate`), застосування міграцій бази даних (`php artisan migrate`) та збірки фронтенд-ресурсів (`npm install` і `npm run build`).

6. Заповнення базовим контентом. Створення першого користувача-адміністратора через Artisan-команду Orchid (`php artisan orchid:admin`) та подальше наповнення лендингу контентом через адміністративну панель.

Уся послідовність детально задокументована у файлі `README.md` репозиторію проєкту, що забезпечує можливість відтворення розгортання іншим розробником.

Стан продакшн-середовища. На момент здачі цієї роботи лендинг розгорнуто на серверній інфраструктурі Херсонського державного університету та доступний за публічною адресою <https://fknfm.kspu.edu/>. Доменне ім'я налаштовано на сервер університету, термінування SSL та обслуговування HTTPS-з'єднань забезпечуються зовнішньою інфраструктурою університету. Скріншоти готового лендингу наведено у Додатку А.

Управління та обслуговування. Після розгортання лендинг функціонує автономно: контейнери MySQL та Apache забезпечують безперервну роботу, всі контентні зміни виконуються через адміністративну панель без зупинки сервісу. Технічне обслуговування включає періодичне оновлення базових образів Docker для отримання патчів безпеки, моніторинг логів через `docker compose logs` та резервне копіювання тому `ksu_fmf_db_data`, що містить весь контент бази даних. У разі необхідності перенесення проєкту на іншу серверну інфраструктуру — це робиться передачею файлів проєкту, файлу `.env` та копії тому бази даних на нове середовище з установленим Docker, без додаткових налаштувань.

Висновок до підрозділу. Контейнеризація через Docker Compose забезпечила можливість швидкого та відтворюваного розгортання лендингу на серверній інфраструктурі університету. Сформована конфігурація з двох сервісів — застосунку з вбудованим веб-сервером Apache та СУБД MySQL — є мінімально достатньою для проєкту такого

масштабу і дозволяє розгорнути повне середовище однією командою. Лендинг розгорнуто у продакшн-середовищі за адресою <https://fknfm.kspu.edu/> та переданий в експлуатацію факультету. Подальше керування контентом виконують представники факультету самостійно через адміністративну панель, що повністю реалізує вимогу автономності, сформульовану на початку роботи. Узагальнення досвіду застосування AI-інструментів на різних етапах розробки лендингу подано в наступному підрозділі.

2.6. Застосування AI-інструментів у процесі розробки: спостереження та практичні висновки

У підрозділі 1.5 розглянуто загальний стан AI-інструментів у сучасній веб-розробці, їх класифікацію та обмеження. У цьому підрозділі узагальнено практичний досвід застосування цих інструментів на конкретних етапах розробки лендингу факультету. Виклад побудовано не як хронологічний звіт, а як систематизація спостережень за категоріями завдань — від найбільш до найменш успішних сценаріїв використання AI.

Інструменти, що застосовувалися. Як було зазначено у підрозділі 1.5, при розробці лендингу використовувалися три AI-інструменти, кожен з яких застосовувався у власній зоні відповідальності:

- **Claude (Anthropic)** [16] як основний асистент генерації серверного коду на Laravel, написання міграцій, налаштування Orchid-екранів, оптимізації Blade-шаблонів та пояснення помилок. Робота з Claude велася у форматі діалогових сесій через веб-інтерфейс claude.ai, з поетапним уточненням завдань і ітеративним доопрацюванням результату.

– **Google Stitch** [20] як експериментальний інструмент для генерації варіантів дизайну окремих секцій сторінки на основі текстових описів і референсних зображень.

– **Gemini (Google)** як допоміжний консультативний інструмент при швидкому пошуку відповідей на технічні питання, порівнянні підходів та перевірці окремих фрагментів коду «другою думкою».

Сценарії, у яких AI-інструменти показали високу ефективність. Найбільш успішними виявилися сценарії з чіткою постановкою завдання, типовою для веб-розробки структурою і відсутністю специфічних залежностей від маловідомих пакетів. До таких сценаріїв належать:

– *Генерація стартового скелета моделей Eloquent та міграцій.* Запит на створення моделі з певним набором полів, з прив'язкою до конкретної таблиці та з визначенням перекладних полів через `astrotomic/laravel-translatable` давав робочий код практично без подальших правок. Claude вільно орієнтується у патернах Laravel, типах полів міграцій, реляціях Eloquent, що відповідає очікуваному рівню володіння найпопулярнішим РНР-фреймворком.

– *Написання сидерів та фікстур.* Створення Seeder-класу для наповнення бази тестовими даними — задача, де якість згенерованого результату виявилася найвищою. Claude коректно генерував не лише структуру коду, а й правдоподібний контент: реалістичні назви програм, варіанти кваліфікацій, опис обов'язкових дисциплін, кар'єрних перспектив. Це дозволило швидко отримати наповнену базу для тестування фронтенду без необхідності ручного заповнення.

– *Пояснення помилок та аналіз стек-трейсів.* При появі помилки під час розробки повідомлення про неї передавалося Claude разом із фрагментом коду, до якого вона стосується. У більшості випадків

модель коректно діагностувала причину помилки, навіть якщо вона була пов'язана зі специфічними особливостями фреймворку — наприклад, із неочікуваною поведінкою middleware, конфліктом маршрутів або помилкою у конфігурації БД.

– *Рефакторинг та оптимізація Blade-шаблонів.* Запити на винесення повторюваної розмітки в окремі компоненти, перегляд послідовності тегів для покращення семантики чи перевірку відповідності розмітки стандартам доступності давали корисні рекомендації. Особливо цінним виявилось використання Claude для генерації ARIA-атрибутів для нестандартних інтерактивних компонентів (акордеону, кастомного select, табів) — модель пропонувала набір атрибутів, який залишалось лише адаптувати під конкретну структуру.

– *Генерація локалізаційних рядків.* Заповнення файлів `lang/uk/common.php` та `lang/en/common.php` парами «ключ — переклад» для повідомлень валідації, написів на кнопках, текстів модальних вікон. Claude генерував коректні переклади для обох мов одночасно, забезпечуючи їх семантичну відповідність.

Ці спостереження загалом узгоджуються з результатами емпіричних досліджень продуктивності AI-асистентів у програмуванні [13]: найбільший вигрaш AI-інструменти дають у задачах генерації шаблонного коду з чіткою специфікацією та обмеженим контекстом.

Сценарії, у яких AI-інструменти потребували значного доопрацювання. Окремо потрібно зафіксувати випадки, де результати роботи з AI були менш якісними і вимагали суттєвого втручання розробника. Зосередження уваги саме на цих випадках важливе тому, що вони характеризують реальні межі застосовності сучасних AI-інструментів у проєктах, що використовують стек поза найпоширенішим мейнстрімом.

Випадок 1. Робота з Laravel Orchid Platform [24]. Найбільш стабільні труднощі виникали при генерації коду для адміністративних екранів на Orchid. Хоча Claude обізнаний з основними концепціями Orchid (Screen, Layout, Field), у деталях API він часто пропонував сигнатури методів, які виглядали правдоподібно, але не існували у поточній версії пакета (14.52), або були застарілими і вилюченими у попередніх релізах. Це призводило до помилок «method does not exist», які доводилося виявляти у процесі тестування і виправляти зверненням до офіційної документації Orchid. Особливо це стосувалося складніших компонентів — `Layout::tabs`, `Layout::accordion`, обробників сортування через drag-and-drop, прив'язок поліморфних attachment-полів. Практичний висновок: при роботі з менш поширеними пакетами екосистеми Laravel результати AI-генерації потребують обов'язкової звірки з офіційною документацією, а не сприйняття «як є». Цей висновок узгоджується з зафіксованими у підрозділі 1.5 контекстними обмеженнями LLM [7; 15] — модель навчена на корпусі загальнодоступного коду, у якому представництво нішевих пакетів об'єктивно менше, ніж основних компонентів фреймворку.

Випадок 2. Інтеграція результатів Google Stitch [20] у проєктний стек. Google Stitch добре справляється з генерацією візуально привабливих варіантів інтерфейсу за текстовим описом, проте практичне використання його результатів у конкретному проєкті виявилось суттєво обмеженим. Stitch за замовчуванням генерує верстку з використанням Tailwind CSS — фреймворку утилітарних класів, який не використовується у клієнтській частині лендингу (як обґрунтовано у підрозділі 2.2, для проєкту обрано чистий CSS3). Спроби прямого перенесення згенерованого HTML вимагали б або підключення Tailwind зі всіма пов'язаними з цим компромісами, або кропіткого ручного переписування кожного класу. Жоден з варіантів не виявився виправданим у контексті проєкту.

Натомість використання Stitch було переосмислене: інструмент застосовувався не як генератор продакшн-коду, а як інструмент швидкого візуального прототипування. Результати генерації служили **референсами** для розуміння компоновки секції, ієрархії візуальних елементів, кольорових та типографічних рішень — і ці референси далі реалізовувалися вручну в Figma з подальшою версткою на чистому CSS. Такий формат використання виявився продуктивним: Stitch генерує варіанти швидше, ніж дизайнер створює макет з нуля, і дозволяє швидко оцінити кілька альтернатив до прийняття дизайн-рішення. Практичний висновок: AI-інструмент, призначений для однієї ролі (генерації коду), може ефективно використовуватися в іншій ролі (візуального прототипування), якщо розробник свідомо переосмислює формат його застосування.

Узагальнені спостереження. На основі досвіду розробки лендингу можна сформулювати кілька узагальнень щодо характеру роботи з AI-інструментами у реальних проєктах.

Перше. AI-інструменти не змінюють характер інженерної задачі, а зміщують **розподіл часу** розробника між її етапами. Замість значної кількості часу на власне написання коду розробник витрачає більше часу на формулювання вимог, читання згенерованого результату, перевірку його коректності та інтеграцію в наявну кодову базу. Сумарний обсяг роботи зменшується, але не зникає, а змінюється за структурою.

Друге. Якість результату значною мірою корелює з якістю постановки завдання [15]. Загальні запити дають загальні рішення, які формально відповідають вимогам, але часто не вписуються в конкретний проєктний контекст. Детальні запити з зазначенням стеку, версій, патернів кодування, очікуваних обмежень дають результат значно вищої якості. Це фактично переносить частину когнітивного навантаження з кодування у формулювання — навичка, яка стає окремим компонентом професійної компетентності розробника.

Третє. AI-інструменти ефективніші у роботі з мейнстрімними технологіями, ніж з нішевими. Чим менш поширений пакет або фреймворк, тим вища ймовірність, що згенерований код містить вигадані сигнатури методів або застарілі API. Для нішевих компонентів стеку (як у випадку з Orchid Platform) AI може використовуватися лише як стартова точка, з обов'язковою наступною звіркою з документацією.

Четверте. AI-інструменти не усувають потреби у технічній експертизі розробника, а змінюють її акцент. Для перевірки коректності згенерованого коду розробник повинен розуміти, як він працює, навіть якщо не написав його власноруч. Здатність читати і критично оцінювати чужий код стає не менш важливою, ніж здатність писати власний.

П'яте. Оптимальною стратегією у поточному стані інструментів є **гібридний підхід**: AI використовується для типових, повторюваних завдань (CRUD, міграції, шаблонний код, переклади, документація), а нетривіальні рішення (архітектура, специфічна бізнес-логіка, інтеграції з нішевими пакетами) залишаються за розробником. У межах цієї роботи такий підхід дозволив отримати функціональний продукт за прийнятний час, водночас зберігаючи повний контроль над якістю і відповідністю кінцевого результату вимогам проєкту.

Висновок до підрозділу. Застосування AI-інструментів на різних етапах розробки лендингу факультету дозволило отримати практичний досвід їх використання у реальному проєкті, що поєднує типові патерни (Laravel, MVC, CRUD) із менш поширеними компонентами (Orchid Platform, astrotomic/laravel-translatable, кастомні Blade-компоненти). Найвищу ефективність AI продемонстрував у завданнях з чіткою структурою — генерації моделей, міграцій, сидерів, рефакторингу шаблонів, пояснення помилок. Найбільші труднощі виникли при роботі з нішевими пакетами екосистеми Laravel та при інтеграції результатів спеціалізованих генераторів інтерфейсів (Google Stitch) у проєкт з нестандартним стеком. Сформульовано п'ять узагальнених

спостережень, що характеризують характер роботи з AI-інструментами у реальних проєктах, і обґрунтовано доцільність гібридного підходу як оптимальної стратегії на поточному етапі розвитку технології.

Висновки до Розділу 2

У другому розділі описано практичну реалізацію лендингу факультету комп'ютерних наук, фізики та математики ХДУ — від проєктування інформаційної архітектури до розгортання у продакшн-середовищі та узагальнення досвіду застосування AI-інструментів на етапах розробки.

1. Спроектовано інформаційну архітектуру лендингу, що включає тринадцять функціональних секцій з послідовністю, орієнтованою на природний порядок сприйняття інформації абітурієнтом. Структура секцій базується на матриці відповідності інформаційних потреб і структурних елементів, сформованій у Розділі 1, та забезпечує повне покриття чотирьох основних груп потреб цільової аудиторії. Окремої уваги при проєктуванні потребували складні функціональні секції — Hero з каруселлю слайдів, освітні програми з табовим перемикачем рівнів та диференційованою навігацією, FAQ з ієрархією категорій, покрокові інструкції для абітурієнтів з диференціацією за освітнім рівнем. Закладено основу дизайн-системи на фірмових кольорах ХДУ, реалізовано двомовність на рівні URL і даних, забезпечено можливість динамічного керування всім контентом через адміністративну панель.
2. Реалізовано клієнтську частину лендингу на чистому стеку HTML5, CSS3 та JavaScript з серверним рендерингом через Blade-шаблони Laravel. Свідомо відхилено застосування SPA-фреймворків (React, Vue) та CSS-фреймворків утилітарних класів (Tailwind) — рішення, обґрунтоване характером продукту як

контентно-орієнтованого ресурсу з обмеженою кількістю інтерактивних елементів. Адаптивність забезпечена за принципом *mobile-first* з трьома контрольними точками; інтерактивні компоненти реалізовано переважно власними засобами із мінімальним залученням зовнішніх бібліотек (виняток — *Embla Carousel* для каруселі *hero*-секції). Окрема увага приділена дотриманню принципів *web accessibility*: семантична розмітка HTML5, послідовна ієрархія заголовків, ARIA-атрибути для нестандартних інтерактивних компонентів, клавіатурна навігація та кольоровий контраст відповідно до WCAG 2.1 рівня AA.

3. Реалізовано серверну частину на *Laravel 12* за класичною архітектурою MVC з ORM *Eloquent* для роботи з базою даних. Спроектowana схема бази включає тематичні таблиці для освітніх програм, FAQ, слайдерів, заявок, налаштувань сайту та SEO-метрик, з винесенням перекладних текстових полів в окрему таблицю *translations* через пакет *astrotomic/laravel-translatable*. Реалізована маршрутизація з підтримкою двох локалей через префікс URL */en/* і middleware *SetLocale*. Обробка контактної форми як основної точки конверсії включає клієнтську та серверну валідацію зі збереженням заявок до бази для подальшої роботи приймальної комісії. Дотримання стандартних практик безпеки *Laravel* — CSRF-захист, екранування *Blade*, параметризовані запити *Eloquent* — дозволило забезпечити захист від поширених вебуразливостей без додаткових налаштувань.
4. Побудовано адміністративну панель на пакеті *Laravel Orchid Platform*, що включає дванадцять функціональних екранів для керування контентом лендингу. Декларативний підхід *Orchid* через концепцію *Screen* дозволив реалізувати повнофункціональну спеціалізовану CMS без написання клієнтського коду для адміністративного інтерфейсу. Реалізована функціональність

забезпечує автономне керування всіма аспектами лендингу представниками факультету: освітніми програмами, каруселями hero-секції, FAQ, заявками, налаштуваннями сайту, SEO-скриптами та перекладами інтерфейсу. Інтеграція системи прикріплення Orchid Attachments дозволила підтримати завантаження файлових матеріалів. Авторизація реалізована через систему ролей Orchid із поточним впровадженням єдиної ролі адміністратора та потенціалом подальшого розширення моделі доступу.

5. Виконано контейнеризацію проєкту через Docker з оркестрацією двох сервісів — застосунку (PHP 8.2 з вбудованим Apache, побудований з кастомного Dockerfile) та СУБД MySQL 8. Конфігурація `docker-compose.yml` є мінімально достатньою для проєкту такого масштабу і дозволяє розгорнути повне середовище однією командою. Розгортання виконано на серверній інфраструктурі Херсонського державного університету; лендинг доступний за публічною адресою <https://fknfm.kspu.edu/> і переданий в експлуатацію факультету. Послідовність розгортання задокументовано у файлі README репозиторію, що забезпечує можливість відтворення на іншому середовищі.
6. Узагальнено досвід застосування AI-інструментів — Claude, Google Stitch та Gemini — на різних етапах розробки лендингу. Зафіксовано п'ять категорій завдань, де AI-інструменти продемонстрували найвищу ефективність: генерація стартового коду моделей і міграцій, написання сидерів, пояснення помилок, рефакторинг шаблонів та генерація локалізаційних рядків. Виокремлено два проблемні сценарії — обмежена обізнаність моделей зі специфічними API нішевих пакетів (Orchid Platform) та невідповідність формату згенерованого Google Stitch коду стеку проєкту з чистим CSS. Сформульовано п'ять узагальнених

спостережень щодо характеру роботи з AI-інструментами: зміщення розподілу часу розробника між етапами, залежність якості результату від якості постановки завдання, нерівномірність ефективності для мейнстрімних і нішевих технологій, збереження потреби у технічній експертизі розробника, доцільність гібридного підходу як оптимальної стратегії на поточному етапі розвитку технології.

Підсумовуючи, лендинг факультету комп'ютерних наук, фізики та математики ХДУ розроблено як функціональний веб-ресурс із повним циклом — проєктування, розробка клієнтської та серверної частин, побудова адміністративного інструментарію, контейнеризація і розгортання у продакшн-середовищі — та переданий в експлуатацію замовника. Узагальнений у підрозділі 2.6 досвід застосування AI-інструментів становить практичний матеріал для рефлексії та подальшого розвитку інженерної практики у сегменті веб-розробки.

ВИСНОВКИ

У межах кваліфікаційної роботи розроблено лендинг факультету комп'ютерних наук, фізики та математики Херсонського державного університету як інструмент комунікації факультету з абітурієнтами та іншими стейкхолдерами. На основі здійсненої теоретичної підготовки та практичної реалізації сформульовано такі підсумкові висновки.

1. *Проаналізовано теоретичні засади розробки лендингів та визначено інформаційні потреби стейкхолдерів факультету.* Уточнено поняття лендингу як спеціалізованого веб-ресурсу з односторінковою архітектурою, орієнтованого на конкретну цільову аудиторію та спрямованого на спонукання її до визначеної дії. Реконструйовано чотири основні групи інформаційних потреб абітурієнтів ІТ-факультету: інформація про освітні програми, умови вступу, навчальне середовище та канали комунікації. На основі аналізу референтної вибірки — лендингів трьох провідних українських ІТ-шкіл (Mate Academy, GoIT, Hillel IT School) — виокремлено сім стійких структурних елементів сучасного освітнього лендингу, що склали основу для проєктування. Водночас зафіксовано принципові відмінності між комерційним і академічним сегментами освіти, що визначають необхідність переосмислення, а не прямого копіювання референтних рішень при перенесенні їх на факультетський лендинг ЗВО.

2. *Обґрунтовано вибір технологічного стеку та підходу до застосування AI-інструментів.* Сформульовано чотири критерії вибору технологій для проєктів освітнього сектору: зрілість компонентів, доступність спільнотної підтримки, можливість автономного управління контентом замовником, переносність розгортання. На основі цих критеріїв сформовано стек з PHP 8.2, Laravel 12, Orchid Platform, MySQL 8, Apache, Blade, astrotomic/laravel-translatable, Vite, Docker. Кожне

технологічне рішення обґрунтовано через зіставлення з альтернативами та аналіз доцільності в контексті проєкту. Окремо розглянуто роль AI-інструментів як допоміжного засобу веб-розробки, зафіксовано їх чотири основні обмеження (якість згенерованого коду, контекстні обмеження, залежність від якості постановки завдання, архітектурні обмеження) та визначено три інструменти для практичного застосування: Claude як основний асистент генерації коду, Google Stitch для експериментів із дизайном, Gemini як консультативний інструмент.

3. *Спроектовано інформаційну архітектуру, дизайн-систему та структуру бази даних лендингу.* Сформовано структуру з тринадцяти функціональних секцій, послідовність яких відповідає природному порядку сприйняття інформації потенційним абітурієнтом. Спроектовано схему бази даних з тематичним розподілом таблиць та винесенням перекладних текстових полів в окрему таблицю через пакет *astrotomic/laravel-translatable*, що забезпечує масштабованість при додаванні нових мов. Закладено дизайн-систему на фірмових кольорах ХДУ з адаптивністю за принципом *mobile-first* і дотриманням принципів *web accessibility* відповідно до WCAG 2.1.

4. *Реалізовано клієнтську частину, серверну частину та адміністративну панель лендингу.* Клієнтську частину побудовано на чистому стеку HTML5 + CSS3 + JavaScript із серверним рендерингом через *Blade*, з обґрунтованою відмовою від SPA-фреймворків та фреймворків CSS-утиліт; інтерактивні компоненти реалізовано переважно власними засобами із семантичною розміткою та підтримкою клавіатурної навігації. Серверну частину реалізовано на *Laravel 12* за архітектурою MVC з підтримкою двомовності на рівні URL і даних, стандартними практиками безпеки фреймворку та обробкою контактної форми через клієнтську й серверну валідацію. Адміністративну панель побудовано на пакеті *Orchid Platform* як спеціалізовану CMS під конкретний проєкт, з дванадцятьма функціональними екранами, що

покривають усі сутності лендингу і забезпечують повну автономність керування контентом представниками факультету без залучення розробника.

5. *Здійснено контейнеризацію та розгортання продукту, узагальнено досвід застосування AI-інструментів.* Контейнеризація реалізована через Docker з оркестрацією двох сервісів (PHP-застосунок з Apache та СУБД MySQL), що забезпечило відтворюваність розгортання на будь-якому Docker-сумісному середовищі. Лендинг розгорнуто на серверній інфраструктурі Херсонського державного університету за публічною адресою <https://fknfm.kspu.edu/> та переданий в експлуатацію факультету. Сформульовано п'ять узагальнень щодо застосування AI-інструментів у реальних проєктах: зміщення розподілу часу розробника між етапами роботи, залежність якості результату від якості постановки завдання, нерівномірність ефективності AI для мейнстрімних і нішевих технологій, збереження потреби у технічній експертизі розробника, доцільність гібридного підходу — поєднання AI-генерації для типових завдань із самостійним розробленням нетривіальних рішень — як оптимальної стратегії на поточному етапі розвитку технології.

Практичне значення одержаних результатів. Розроблений лендинг є функціональним веб-ресурсом, переданим в експлуатацію факультету комп'ютерних наук, фізики та математики ХДУ. Адміністративна панель забезпечує автономність керування контентом представниками факультету без залучення розробника. Узагальнені у роботі спостереження щодо застосування AI-інструментів на різних етапах веб-розробки становлять матеріал для рефлексії, корисний для розробників-початківців та студентів профільних спеціальностей при формуванні власних практик роботи з сучасними інструментами.

Перспективи подальшого розвитку. Поточна реалізація лендингу залишає простір для подальшого розширення функціональності у

кількох напрямках. По-перше, поетапне наповнення розділів реальним контентом, що наразі перебуває у процесі заміни шаблонних матеріалів, — насамперед це стосується історій успіху випускників, відгуків студентів та інформації про компанії-партнери. По-друге, розширення моделі ролей доступу через впровадження окремих ролей редактора контенту та модератора заявок з гранулярними дозволами. По-третє, інтеграція з зовнішніми системами — зокрема, інтеграція форми заявок з CRM-системою приймальної комісії для уникнення ручного перенесення даних. По-четверте, систематичне використання SEO-метрик через підключення Google Analytics та Search Console для отримання даних про реальну поведінку відвідувачів, що дозволить ітеративно вдосконалювати структуру сторінки на основі емпіричних даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Про вищу освіту : Закон України від 01.07.2014 р. № 1556-VII.
URL: <https://zakon.rada.gov.ua/laws/show/1556-18> (дата звернення: 23.05.2026).
2. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. [Чинний від 2016-07-01]. Київ : ДП «УкрНДНЦ», 2016. 16 с. (Інформація та документація).
3. Каленюк І. С., Куклін О. В. Розвиток вищої освіти та економіка знань : монографія. Київ : Знання, 2012. 343 с.
4. Манжура О. В. Маркетинг освітніх послуг : навч. посіб. Київ : Центр учбової літератури, 2020. 248 с.
5. Ash T., Ginty M., Page R. Landing Page Optimization: The Definitive Guide to Testing and Tuning for Conversions. 2nd ed. Indianapolis : Wiley, 2012. 480 p.
6. Duckett J. HTML and CSS: Design and Build Websites. Indianapolis : Wiley, 2011. 490 p.
7. Fan A., Gokkaya B., Harman M., Lyubarskiy M., Sengupta S., Yoo S., Zhang J. M. Large Language Models for Software Engineering: Survey and Open Problems. *arXiv preprint*. 2023. arXiv:2310.03533. URL: <https://arxiv.org/abs/2310.03533> (дата звернення: 23.05.2026).
8. Freeman R. E. Strategic Management: A Stakeholder Approach. Cambridge : Cambridge University Press, 2010. 292 p.
9. Haverbeke M. Eloquent JavaScript: A Modern Introduction to Programming. 4th ed. San Francisco : No Starch Press, 2024. 456 p.
10. Krug S. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability. 3rd ed. Berkeley : New Riders, 2014. 216 p.
11. Marcotte E. Responsive Web Design. 2nd ed. New York : A Book Apart, 2014. 153 p.

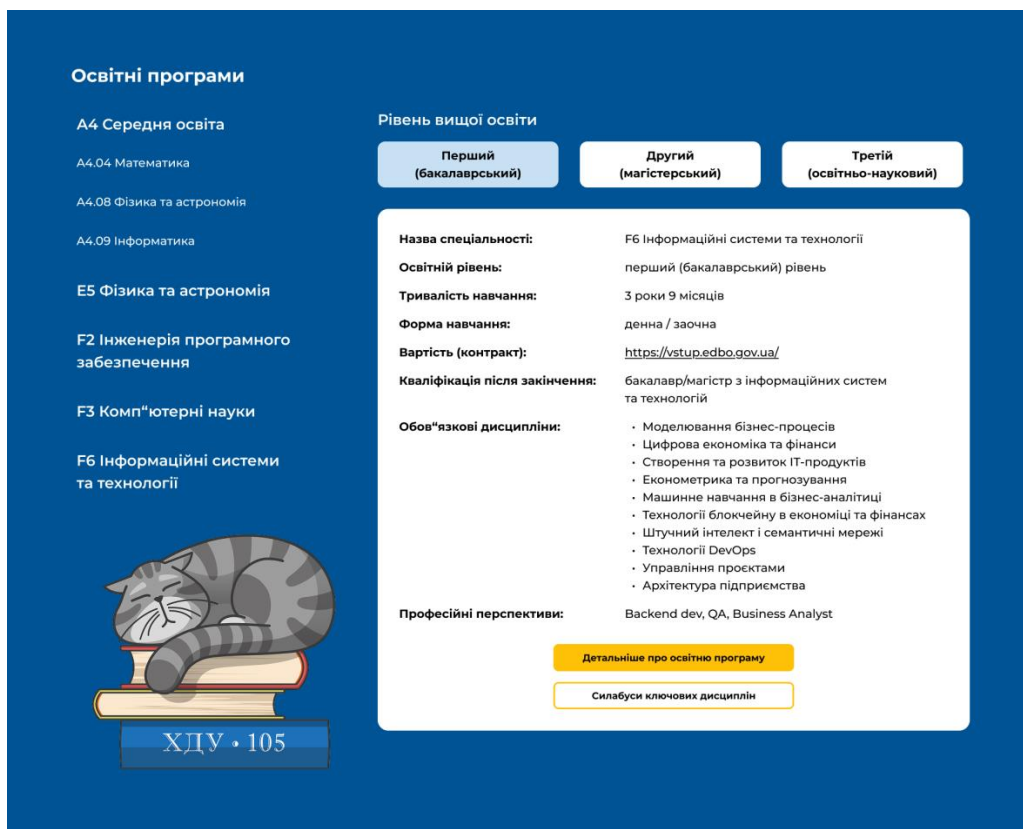
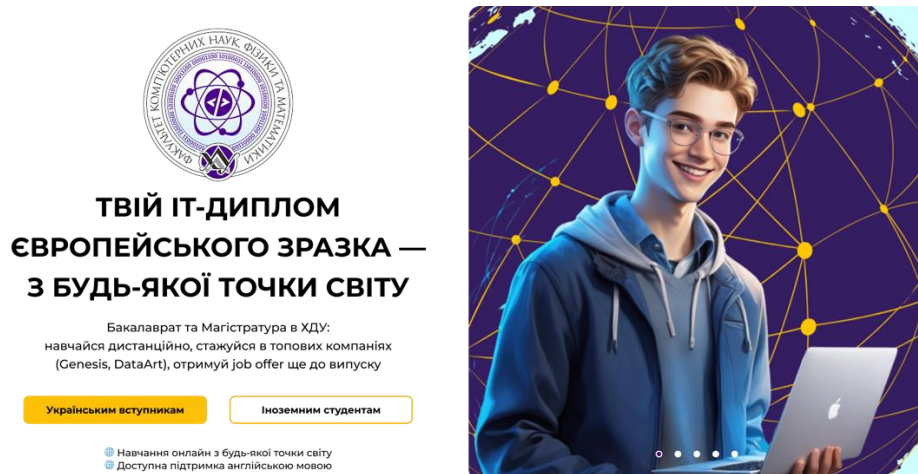
12. Nielsen J. Usability Engineering. San Francisco : Morgan Kaufmann, 1993. 362 p.
13. Peng S., Kalliamvakou E., Cihon P., Demirer M. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. *arXiv preprint*. 2023. arXiv:2302.06590. URL: <https://arxiv.org/abs/2302.06590> (дата звернення: 23.05.2026).
14. Stauffer M. Laravel: Up and Running: A Framework for Building Modern PHP Apps. 3rd ed. Sebastopol : O'Reilly Media, 2023. 569 p.
15. Vaithilingam P., Zhang T., Glassman E. L. Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models. *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*. New York : ACM, 2022. P. 1–7. DOI: <https://doi.org/10.1145/3491101.3519665>.
16. Anthropic. Claude Documentation. URL: <https://docs.claude.com> (дата звернення: 23.05.2026).
17. Astrotoomic. Laravel Translatable Package. URL: <https://docs.astrotoomic.info/laravel-translatable/> (дата звернення: 23.05.2026).
18. Docker Inc. Docker Documentation. URL: <https://docs.docker.com> (дата звернення: 23.05.2026).
19. GoIT Global. Курси програмування з подальшим працевлаштуванням. URL: <https://goit.global/ua/> (дата звернення: 23.05.2026).
20. Google. Stitch — Design with AI. URL: <https://stitch.withgoogle.com/> (дата звернення: 23.05.2026).
21. Hillel IT School. Курси IT-технологій. URL: <https://ithillel.ua/> (дата звернення: 23.05.2026).
22. Laravel. Documentation. URL: <https://laravel.com/docs/12.x> (дата звернення: 23.05.2026).

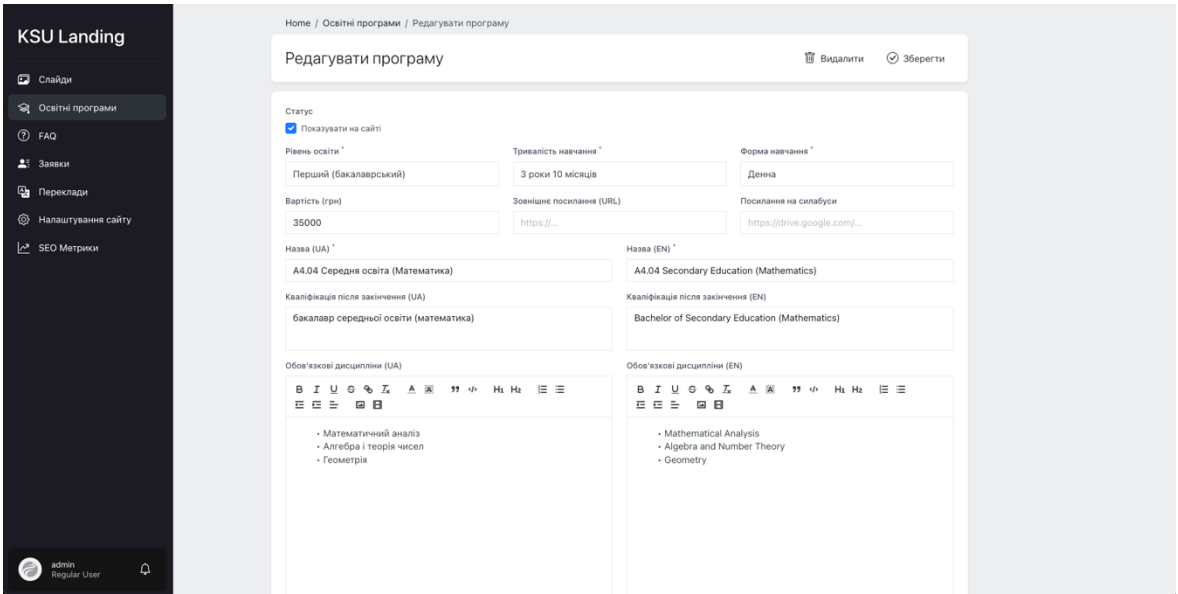
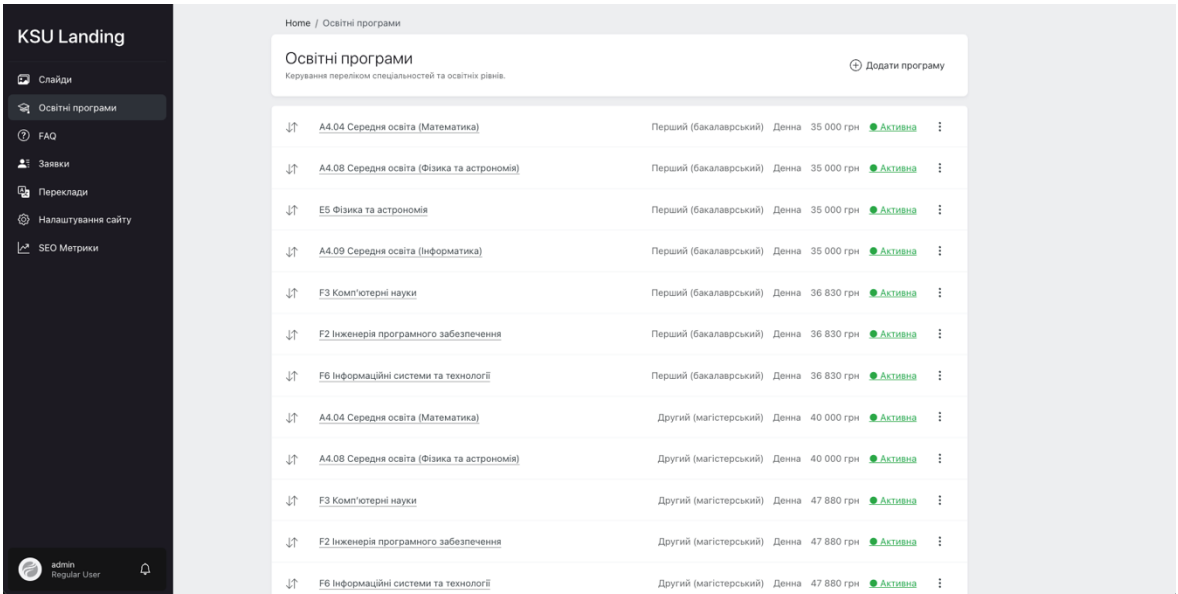
23. Mate Academy. IT-курси з гарантією працевлаштування. URL: <https://mate.academy/> (дата звернення: 23.05.2026).
24. Orchard. Laravel Orchard Platform Documentation. URL: <https://orchid.software/en/docs> (дата звернення: 23.05.2026).
25. Oracle Corporation. MySQL 8.0 Reference Manual. URL: <https://dev.mysql.com/doc/refman/8.0/en/> (дата звернення: 23.05.2026).
26. Vite. Next Generation Frontend Tooling. URL: <https://vitejs.dev/> (дата звернення: 23.05.2026).
27. W3C Web Accessibility Initiative. Web Content Accessibility Guidelines (WCAG) 2.1. W3C Recommendation. URL: <https://www.w3.org/TR/WCAG21/> (дата звернення: 23.05.2026).

ДОДАТКИ

Скріншоти готового лендингу

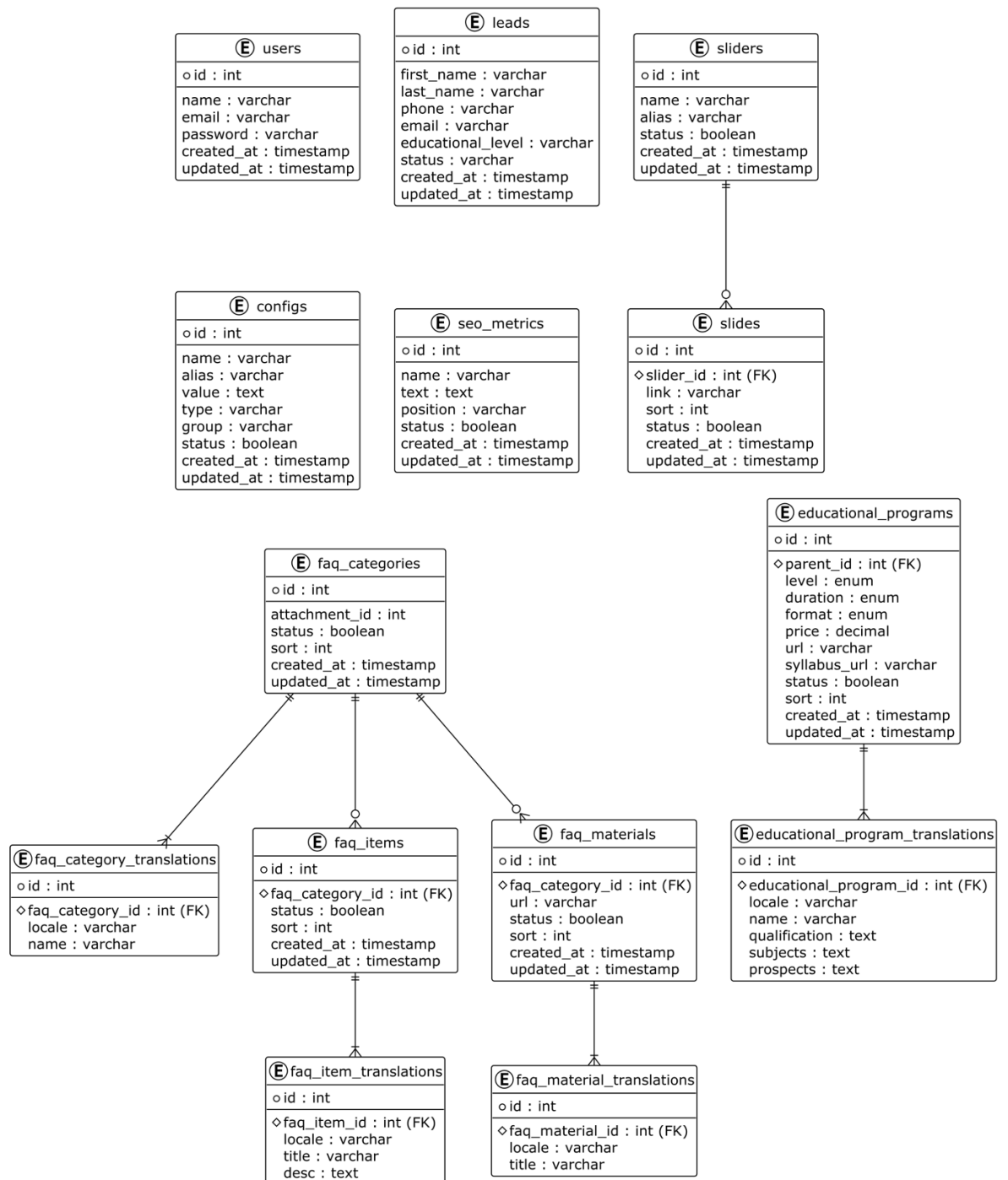
У цьому додатку наведено скріншоти готового лендингу факультету комп'ютерних наук, фізики та математики ХДУ, розгорнутого за адресою <https://fknfm.kspu.edu/>.





Структура бази даних лендингу

Концептуальна схема основних таблиць бази даних лендингу та їх зв'язків наведена нижче. Перекладні поля винесено в окрему таблицю translations через пакет `astrotomic/laravel-translatable`.



Програмний код

У цьому додатку міститься посилання на відкритий репозиторій GitHub, який містить повний вихідний код розробленого лендингу та ілюструє технічні рішення, описані у Розділі 2.

Посилання на репозиторій: <https://github.com/helxer/fizmat-landing>

Приклади запитів до AI-інструментів та отриманих результатів

У цьому додатку наведено три репрезентативні приклади взаємодії з AI-інструментами, що ілюструють спостереження, узагальнені у підрозділі 2.6: успішний випадок, випадок з контекстними обмеженнями, та випадок переосмислення формату використання інструменту.

Г.1. Приклад успішного запиту: генерація сидера

Запит до Claude (Anthropic): «Створи Laravel Seeder для таблиці `educational_programs` з полями: `parent_id` (nullable int), `level` (enum), `duration` (enum), `format` (enum), `price` (decimal), `url` (string), `syllabus_url` (string), `status` (boolean), `sort` (integer). Перекладні поля через `astroatomic/laravel-translatable` для таблиці `educational_program_translations`: `name` (string), `qualification` (text), `subjects` (text), `prospects` (text). Створи 12 записів — 7 бакалаврських і 5 магістерських програм у галузях інженерії ПЗ, комп'ютерних наук, інформаційних систем, фізики, математики, освіти за предметом. Обидві мови — uk та en»

Результат. Claude згенерував повноцінний клас `EducationalProgramSeeder`, що використовує `Model::create()` з масивом полів та секцією `'translations'` => [...] для двомовного контенту. Сидер містив реалістичні назви програм («Інженерія програмного забезпечення», «Прикладна інформатика», «Середня освіта за предметами фізика та інформатика»), осмислені описи перелічених обов'язкових дисциплін та орієнтовні професійні перспективи випускників. Згенерований код вимагав мінімальних доопрацювань — додавання `truncate()` на початку

методу `run()` для уникнення дублювання при повторному запуску і коригування орієнтовних цін відповідно до реальних розцінок ХДУ.

Г.2. Приклад запиту з контекстними обмеженнями: екран Orchid

Запит до Claude (Anthropic): «Створи Orchid Screen для редагування освітньої програми з табами української та англійської мови (`Layout::tabs`), полями `name`, `qualification (textarea)`, `subjects (textarea)`, `prospects (textarea)` для кожної мови, та мовнонезалежними полями `parent_id (select)`, `level (select)`, `duration (select)`, `format (select)`, `price`, `url`, `syllabus_url`, `status (checkbox)`, `sort`.»

Результат і його обмеження. Claude згенерував клас `EducationalProgramEditScreen` з методами `query()`, `layout()`, `commandBar()`. Структура коду була правильною на загальному рівні, проте містила кілька помилок: метод `TextArea::make()` був написаний як `Textarea::make()` (каamelкейс іншого регістру); конструкція `Layout::tabs([...])` була застосована з прив'язкою до перекладних полів за допомогою методу `->lang('uk')`, який у поточній версії Orchid 14.52 вже не існує — замість нього треба використовувати `Group` з елементами для кожної мови або інше рішення. Помилки виявлено лише після запуску сторінки і отримання повідомлень типу «Method does not exist». Виправлення зайняло близько години роботи з офіційною документацією Orchid 14 (<https://orchid.software/en/docs/>), що підтверджує спостереження з підрозділу 2.6 щодо обмеженої обізнаності AI-моделей з API нішевих пакетів.

Г.3. Переосмислення формату використання: Google Stitch як референс

Запит до Google Stitch: «Generate a hero section for a Ukrainian university IT faculty landing page. Two call-to-action buttons: one for local Ukrainian applicants, one for international students. Modern, professional, with purple accent (#4E008E). Include a background image carousel area on the right side. Mobile-first responsive design.»

Результат і його використання. Stitch згенерував кілька варіантів дизайну hero-секції з різними композиційними рішеннями: класичний макет «текст ліворуч + зображення праворуч», варіант з повноекранним фоновим зображенням, варіант з градієнтним фоном без зображення. Кожен варіант супроводжувався згенерованим HTML кодом із використанням Tailwind CSS. Оскільки проєкт використовує чистий CSS3 без Tailwind (обґрунтування у підрозділі 2.2), пряме перенесення коду було неможливим. Однак візуальні рішення Stitch — пропорції елементів, кольорові акценти, ієрархія типографіки — стали корисними референсами при створенні фінального макета у Figma. Зокрема, варіант із розділенням на дві колонки (текст 50% / карусель 50% на десктопі) із вертикальним стеком на мобільній версії був прийнятий як основа hero-секції, та переверстаний на чистому CSS. Це переосмислення підтверджує висновок з підрозділу 2.6: AI-інструмент, призначений для генерації коду, може ефективно використовуватися в ролі візуального прототипування при свідомому переосмисленні формату його застосування.