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

ПРОЄКТУВАННЯ ТА РОЗРОБЛЕННЯ СЕРВІСНОЇ АРХІТЕКТУРИ УПРАВЛІННЯ БІЗНЕС-ПРОЦЕСАМИ УНІВЕРСИТЕТУ. ПРОГРЕСИВНИЙ МОБІЛЬНИЙ ДОДАТОК

Кваліфікаційна робота (проєкт)
на здобуття ступеня вищої освіти "магістр"

Виконав: здобувач 2 курсу, 241М групи
Спеціальності 121 «Інженерія програмного забезпечення»
Освітньо-професійної програми «Інженерія програмного забезпечення» другого (магістерського) рівня вищої освіти
Шкворець Владислав Владленович
Керівник: доктор педагогічних наук, кандидат фізико-математичних наук
Співаковський Олександр Володимирович
Рецензент: доктор педагогічних наук, кандидат фізико-математичних наук, професор
Кузьменков Сергій Георгійович

Херсон – 2021

ЗМІСТ

ВСТУП

Актуальність теми. З роками попит на програмне забезпечення для мобільних пристроїв постійно зростає через збільшення ринку смартфонів. Розробники мобільних пристроїв мають право приймати різні архітектури або стратегії розвитку, які включають нативні програми, мобільні веб-програми, гібридні програми та нові прогресивні веб-програми (Progressive Web App). PWA, що поєднує в собі особливості стратегій розробки нативних та веб-додатків, стала кращою альтернативою іншим підходам до розробки завдяки додатковим перевагам, таким як можливість роботи в автономному режимі, синхронізація у фоновому режимі тощо, незважаючи на декілька проблем, які були виявлені щодо ефективності PWA.

Ця магістреська робота має на меті провести порівняльне дослідження існуючих архітектур мобільного розвитку з використанням методики систематичного огляду літератури (SLR), виконати порівняння функцій архітектур Native, Hybrid та PWA і, нарешті, аргументувати архітектуру розробки PWA на основі порівнянь.

Все більше людей розпочинають відкривати для себе веб-додатки та нативні програми для управління безнес процесами. І для того щоб университет продовжив ставати більш привабливим для студентів, йому потрібен власний веб-додаток студентоорієнтований для управління бізнес-процесами. Розробка такого додатку є спільним проєктом, що виконується в Херсонському державному університеті. На різних задачах розробки були задіяні магістранти Сенчишен Д., Романенко В., Валяєв К., Лукінов Г., Коломієць О. [44,45,46,47].

Метою роботи є розроблення сервісної архітектури управління бізнес-процесами університету.

Об’єктом є інтерфейси мобільних додатків управління бізнес-процесами університету.

Предметом роботи є мобільний додаток KSU24.

Завдання роботи:

1. Аналіз аналогів
2. Обґрунтування вибору технологій за результатами порівняльного аналізу
3. Проєктування та розробка системи

Структура роботи. Робота складається зі вступу, трьох розділів, висновків та списку використаних джерел та додатків.

РОЗДІЛ 1
АНАЛІЗ БІЗНЕС-ПРОЦЕСІВ УНІВЕРСИТЕТУ

1.1 Загальні бізнес-процеси університета

Вищі навчальні заклади мають величезне навантаження, коли йдеться про те, щоб студенти та персонал мали доступ до всіх ресурсів, необхідних для здобуття вищої освіти.

Проте, незважаючи на неймовірний прогрес технологій за останні роки, більшість наукових кіл все ще вдаються до ручних робочих процесів, щоб виконати значну частину цієї адміністративної роботи. Це призводить до купи документообігу та роз'єднаних процесів, у яких легко виникають помилки, документи зникають або студенти використовують обхідні шляхи, що може мати негативні наслідки у випадку таких процесів, як подання заяв про фінансову допомогу. Не тільки це, але талановитий персонал витрачає незліченну кількість годин на такі шаблонні справи, як введення даних та електронні листи, що погано для рівня задоволення роботою.

Рішення: управління бізнес-процесами (BPM - Business Process Management System/Tool). BPM – це управління бізнес-процесами. Воно відноситься до практики оцифровки, проактивного моніторингу та оптимізації послідовності завдань, які необхідно виконати для досягнення певної бізнес-мети. На щастя для бізнесу та вищих навчальних закладів програмне забезпечення для автоматизації бізнес-процесів може зробити це набагато простіше.

Ключові бізнес-процеси у вищій освіті. Багато в чому вищі навчальні заклади, такі як коледжі та університети, подібні до великих корпорацій, і всі основні бізнес-процеси, які зустрічаються на великих підприємствах, також діють у вищих навчальних закладах.

Однак, крім повсякденних бізнес-процесів, які є рушійною силою кожного бізнесу, таких як замовлення на закупівлю, залучення

персоналу та запити на вихідні документи, бізнес-процеси вищої освіти також включають такі процеси, як вступ та реєстрація студентів, обробка заяв про фінансову допомогу, затвердження навчальних планів, складання графіків та десятки інших адміністративних процедур, що стосуються наукових кіл.

Розглянемо детальніше деякі з процесів, що відбуваються у більшості вищих навчальних закладів [39,42]:

- Загальні адміністративні робочі процеси
- Реєстрація студентів,
- Координуючи різні факультети та кафедри
- Реєстрація студентів з особливими потребами, ведення обліку відвідувань та лікарняних листів, відстрочок іспитів тощо.
- Координація вручення дипломів, нагород, академічних та спортивних призів,
- Виділення та адміністрування бібліотек та лабораторій,
- Планування зустрічей, семінарів, гостьових лекцій тощо.
- Адміністрування систем управління навчанням в Інтернеті (LMS),
- Керування іспитами, оцінюванням тощо.
- Культурно-спортивні та легкоатлетичні заходи
- Інформаційні технології
- Адміністрування комп’ютерної лабораторії,
- Запити на апаратне та програмне забезпечення,
- Керування засобами друку та копіювання та обліковими записами студентів
- Налаштування електронної пошти в корпусу, Wi-Fi на території корпусу,
- Оновлення програмного забезпечення тощо.
- Управління інформацією

- Збір інформації, ведення та реєстрація документів,
- Оновлення баз даних (наприклад, академічні записи студентів),
- Управління людськими ресурсами

Це далеко не вичерпний перелік бізнес-процесів, які діють у вищому навчальному закладі. Кожен із пунктів у цьому списку являє собою робочий процес і ймовірно, форму збору даних або шанс, що більшість із них все що виконується вручну. Завдяки автоматизації бізнес-процесів великі сегменти та цілі робочі процеси можна автоматизувати, заощаджуючи тисячі годин, витрачених раніше на повторювані адміністративні завдання.

Автоматизація може допомогти впорядкувати трудомісткі адміністративні процеси, зменшивши адміністративний тягар, з яким стикаються вищі навчальні заклади. Це безпосередньо перетворюється на більш ефективні операції, що, в свою чергу, здійснює економію часу, грошей та сприяє більш великій кількості залучених студентів до життя університету [36,37].

Ось кілька прикладів переваг BPM у вищій освіті:

1. Управління бізнес-процесами дозволяє набагато швидше залучати студентів, дозволяючи їм зосередитися на влаштуванні та налаштуванні на успіх у навчанні, а не витрачати години на нудні адміністративні процеси.

2. Покращення адміністративних процесів призводить до скорочення часу очікування та більшого доступу до актуальної інформації, покращуючи досвід студентів в цілому та надаючи студентам ресурси, необхідні для успішного навчання. Це особливо актуально щодо прискорення розгляду заяв.

3. Автоматизація процесів вищої освіти дозволяє викладачам зосередитись на освіті та наукових дослідженнях, а не на

адміністративних завданнях, покращуючи задоволеність роботою та одночасно підвищуючи якість навчального досвіду студентів.

4. Використання даних, отриманих від оцифрованих процесів, дає змогу виміряти робочі процеси, визначити больові точки та вузькі місця та здійснити подальше підвищення ефективності та функіонування навчального закладу.

5. Автоматизація облікових процесів у вищій освіті, таких як схвалення рахунків-фактур та робочі процеси схвалення звітів про витрати, може заощадити багато грошей, запобігти помилкам та шахрайству.

Отже, автоматизація бізнес-процесів може мати величезне значення у вирішенні багатьох адміністративних завдань, що беруть участь у повсякденному функціонуванні вищого навчального закладу. Автоматизації підлягають як процеси, пов'язані з управлінням матеріальними та людськими ресурсами, так і пов'язані безпосередньо з освітнім процесом. Важливим завданням є інтеграція різних процесів у єдиній системі та можливість оперативного та безпечного доступу до різних даних.

1.2 Створення онтології бізес процесів

Щоб створити більш надійну основу для архітектурної пропозиції бізнес процесів (БП) авчального закладу, ми створили онтологію БП. Ми не будемо наводити технічні деталі тут, але представимо кінцевий результат наших зусиль у вигляді діаграми класів UML (додаток Б), яка детально описує структура БП університету.

Ми починаємо з визначення макропроцесів:

Макропроцес1 (макрос1): Ланцюг доданої вартості або група процесів університету, які визначають мету бізнесу; вони охоплють всі види діяльності від запитів клієнтів до результату.

Макропроцес2 (Macro2): Група процесів, які бізнес виконує для розвитку нових можливості, необхідні для підвищення його конкурентоспроможності.

Макропроцес3 (Макро3): бізнес-планування або група процесів, які визначають майбутнє бізнесу у вигляді стратегій, планів та програм.

Macroprocess4 (Macro4): Група процесів, які керують ресурсами, необхідними інші функції: фінансова, людська, інфраструктурна та інші. Онтологія також показує, що Macro2 і Macro3 мають певну спеціалізацію справ.

Business Process Management and Optimization (BPMO) описує багату модель бізнес-процесів, як того вимагає спільнота Business Process Management (BPM), використовуючи онтологічні описи для фіксації робочого процесу та організаційних проблем у єдиній та розширюваній формі, а також повторно використовує результати досліджень семантичних веб-служб для опису діяльності взаємодії.

Використання BPMO має різні переваги. По перше, BPMO надає вичерпні семантичні примітки для бізнес-процесів, які можна використовувати для автоматизованого виведення на бізнес-рівні, одночасно полегшуючи переклад на рівень виконання. По-друге, BPMO надає посилання від процесу до організаційних аспектів, які можна моделювати незалежно для різних доменів. По-третє, BPMO можна використовувати для перевірки на семантичному рівні обмежень, що застосовуються до робочого процесу або певних процесів. Нарешті, BPMO полегшує моделювання нових (або посередницьких) процесів на основі існуючих, а також відкриття служб для цільової діяльності.

BPMO полегшує семантичну сумісність, моделюючи дії взаємодії, використовуючи описи Semantic Web Services (SWS) входів, виходів та операцій. Ці дії використовують онтологічно визначені дані для потоку даних, а також використовують переваги семантичних відображень, наданих посередниками даних BPMO.

Діаграми BPMO можна створювати за допомогою практичних і вільно доступних інструментів за допомогою WSMO Studio. Перевагою є те, що модельєр BPMO WSMO Studio автоматично генерує екземпляри BPMO з діаграми робочого процесу і дозволяє легко посилатися на екземпляри онтології та описи послуг. BPMO використовує WSML-Flight як мову представлення, яка може бути використана разом з резонатором IRIS для виконання перевірки екземпляра та запитів.

З точки зору бізнесу, бізнес-аналітики можуть виконувати семантично включені запити безпосередньо та однорідно щодо бізнес-контексту та діяльності бізнес-процесу. Запити можуть бути розширені до пунктів призначення та джерел перекладу BPMO протягом усього життєвого циклу процесу від створення до розгортання, моніторингу та виконання. Таким чином, полегшується повторне використання в бізнесі/IT-поділі та досягається велика масштабованість завдяки збільшенню автоматизації, що підтримується міркуваннями на основі онтології.

Існує значна робота з обговорення перекладу та невідповідностей між Business Process Model and Notation (BPMN) та Business Process Execution Language (BPEL) (наприклад, [17], [18]) та, загалом, між нотаціями робочого циклу що проінформувало про реалізацію концепцій та атрибутів BPMO, особливо що стосується можливості перекладу конструкцій BPMO з нотацій, таких як BPMN, та на такі мови, як BPEL. Наприклад, розроблені декілька перекладачів (наприклад, [15]) у рамках проекту SUPER, які використовують відповідні види шаблонів робочих процесів у BPMO, щоб уникнути використання робочих процесів з ациклічними циклами та несинхронізованими гілками. Одна з основних відмінностей від існуючих стандартів до BPMO полягає в тому, що було використано онтології та розширення для підтримки семантичних веб - служб [38].

OWL-S9, онтологія SWS, подана до W3C, містить опис робочого процесу процесу на основі семантики (тобто модель процесу), яка служить тій же меті, що і BPMO; проте ця модель не дуже багата. Як зазначено в [3], існує ряд конструкцій з BPEL, таких як умови, синхронізація та зовнішні (на основі подій) вибори та обробники, які не можна виразити у OWL-S [42].

Підхід семантичного Інтернету, представлений у [11], має подібні цілі з нашим підходом із використанням BPMO, оскільки автори стверджують, що синтаксичний підхід, наданий BPEL, має недоліки, які обмежують його здатність забезпечувати безперебійну сумісність. Вони пропонують використання технологій на основі семантики (OWL-S) для підтримки автоматизованого виявлення послуг, налаштування та семантичного перекладу для процесів на основі BPEL; однак їх анотації щодо послуг та даних відокремлені від мови потоку управління. Натомість BPMO надає семантично анотовані конструкції потоку управління разом із семантичними описами даних та послуг. Крім того, робочий процес BPMO включає посередництво семантичних даних за допомогою завдань посередництва та посередників даних, які можуть посилатися на правила відображення та послуги посередництва.

У [25] представлений підхід під назвою «Шаблони семантичних процесів» (SPT), який надає семантичні розширення специфікації робочого процесу, сумісного з BPEL (на основі XML). Семантичний шаблон використовується для кожної діяльності у визначенні процесу, щоб приєднати поняття з даної онтології до входів, виходів та операцій. SPT відрізняється від BPMO тим, що це підхід знизу вгору, прив'язаний до певного стандарту виконання (BPEL), а конструкції потоку управління не мають онтологічного представлення.

Отже, представлена методологія дозволяє інтегрувати дизайн від архітектури до бізнес-логіки, але вона поки що формально не інтегрована до програм підтримки BP.

РОЗДІЛ 2
ОБҐРУНТУВАННЯ ВИБОРУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОРІВНЯННЯ ХАРАКТЕРИСТИКИ ПРОГРАМНИХ КОМПОНЕНТІВ

2.1 Обґрунтування вибору фрейморку за результатами співставлення фрейморків Flask та Django.

У цьому пункті ми обговоримо ключові відмінності між Django та Flask.

Django - це безкоштовний, але дійсно просунутий фреймворк. Його охоче обирають для роботи та навчання як професіонали, так і аматори. Чому він такий популярний? Він пропонує доступ до ефективних систем кешування, автоматично створювану панель адміністрування тощо. Цікаво, що з її допомогою було створено багато додатків, відомих у всьому світі. Приклади цього включають Instagram та Pinterest. Тож кожен у Django знайде те, що йому потрібно [5,6].

Flask - це невелика структура, яка називається мікропрограмою. Він не має передових інструментів та бібліотек, але все ж його структура легка та модульна. Завдяки цьому Flask здобув в очах ентузіастів програмування на Python свою нішу і дуже часто використовується замість Django, наприклад, для проектування та створення менш складних програм та систем. Flask використовують на Linkedin [9,13].

Відмінності між фреймворками Django та Flask

Django - це великий і вдосконалений фреймворк з розгорнотою структурою. Користувач отримує все необхідне. Він ідеально підходить для більшості систем. Єдиним недоліком може бути складна модифікація конкретних модулей. [1,3]

Flask, в свою чергу, є мікрофреймворком. Цікаво, що він був створений як першо квітневий жарт, метою якого було пародіювати

фреймворк Bottle. Спільнота розробників, однак, виявила, що це дійсно хороше вирішення багатьох проблем, і з часом воно стало одним з найпопулярніших фреймворків Python у світі. Хоча він вимагає більшої конфігурації, ніж Django, він не має нав'язаної структури, що покращує гнучкість роботи програміста [6,10].

Гнучкість та маштабування

Яка структура є більш гнучкою, Flask або Django? Багато програмістів відповідають: Flask більш гнучкий, ніж Django, і крапка. Але чи так це насправді?

У 2012 році різниця була дійсно великою, тоді у Flask можна було вибрати різні бібліотеки для сеансів, форм, шаблонів, користувацької моделі користувача / Angular. Крім того, Django тепер набагато гнучкіший, ви можете налаштувати модель користувача, ми більше не змушені використовувати бекенд для створення інтерфейсу [14].

То де ж ця гнучкість Flask? Напевно, тільки у обрання бази даних. У Django вибір бази даних noSQL робить 80% фреймворку марним. Але у вас є ще 20% дійсно хороших речей, які ви можете використовувати точно так само, як і використання Flask .

Ви, напевно, задаєтесь питанням: "Чи повинен я використовувати Django лише для використання 20% його коду?" Я думаю так, тому що 90% інших проектів все одно будуть використовувати Django, тож вам не доведеться вивчати інший фреймворк.[19]

Безбека фраймворків

Джанго тут безперечний переможець. Як великий фреймворк, він контролює більшість аспектів веб-додатків, дозволяючи йому застосовувати стандарти безпеки, навіть якщо розробники не знайомі з ними.

Інша річ, що надбудови Django набагато потужніші, ніж надбудови Flask, а отже, вони також можуть краще контролювати аспекти безпеки. Розглянемо рівень безпеки аутентифікації. Наприклад в

середовище Django існує відоме доповнення Django Simple JWT: на встановлення та налаштування якого потрібно 5 хвилин і він є стандартом в автентифікаціїнії галузі [5].

З іншого боку, Flask не може запропонувати жодних гарантій безпеки. Ти там один. Багато чого доведеться вчитися самостійно. Ви щось зламаєте під час навчання? Безумовно.[16]

Після детального аналізу переваг та недоліків фреймворків ми можемо обрати який нам підходе найбільш. Звертаючи увагу на те що наша архітектура повинна мати можливість к масштабуванню та велику ступінь захисту ми обрали Django.

2.2 Порівняння PWA та нативного додатку

Як вони написані: різні мови для різних цілей

У той час як нативні програми написані для роботи на мобільних пристроях, PWA - для роботи всередині веб-браузера. Нативні програми розробляються з використанням мов програмування кожної платформи (Objective-C та Swift для iOS та Java для Android), тоді як PWA використовує HTML, CSS та JavaScript.

Вартість розробки:

Створення прогресивного веб-додатка дешевше, ніж розробка власного додатка. У випадку нативного додатка вам доведеться вивчити мову та створити версію для кожної платформи. Це означає, що вам потрібні принаймні дві версії для iOS та Android, а також ресурси для підтримки та оновлення кожної версії. Залежно від мети та складності програми, це вимагає багато часу та грошей.

Прогресивний веб-додаток швидше створюється та оновлюється. Ви можете мати єдину базу кодів для різних платформ, а не тільки для двох популярних платформ. Замість того, щоб розробляти додаток з нуля, ви можете налаштувати свій поточний веб-сайт за допомогою

таких інструментів, як Google Lighthouse. Завдяки адаптивному дизайну вам потрібна лише одна версія програми, і вона буде відображатися однаково на всіх пристроях.[32]

Розповсюдження мобільних додатків

З власним додатком, окрім розробки окремих версій для різних платформ, вам також доведеться надсилати їх у різні магазини додатків. Найпопулярнішими є App Store Apple і Play Store Android, але є також App Store Amazon, Windows Store тощо. Для кожного з цих магазинів вам доведеться пройти певні вимоги, щоб опублікувати їх. Іноді за реєстрацію облікового запису розробника навіть доводиться платити певну плату.

З іншого боку, PWA обходить громіздкі вимоги App Store. Користувачам потрібен лише веб-браузер та URL-адреса. Багато функцій PWA підтримуються популярними браузерами, такими як Chrome, Safari, Firefox та Edge. [4]Це полегшує охоплення програми великою аудиторією за короткий час. Ви можете легко розгорнути оновлення, не чекаючи підтвердження. Це робить PWA набагато зручнішим як для вас, так і для користувачів.

Вимоги таких магазинів як Apple store та Google store не дозволяють публікувати програми низької якості. Публікація програми може підвищити вашу надійність та надасть користувачам більше впевненості у доступі до вашої програми, а не посилання на URL - адресу. Веб-магазини також можуть просувати ваш бізнес: розміщення в магазині App Store може швидко пришвидшити розвиток бренду та продажів.

Сторінки нативного додатка не можна індексувати та рахувати у пошуковій системі. Користувачі можуть знайти ваш додаток у магазині додатків або на веб-сайті магазину додатків. Існує ряд факторів, що впливають на виявлення додатків, і ви можете допомогти людям

швидше знайти ваш додаток за допомогою Оптимізації магазинів додатків (ASO).

ASO-це процес підвищення вашого рейтингу результатів пошуку в магазині. Це передбачає дослідження ключових слів, написання ефективного заголовка та META-опису, створення хороших скріншотів, використання відповідної категоризації або звернення до третіх сторін для завантаження та огляду тощо. Однак все це додає додатковий час та витрати на доставку вашого додатка до ринку.

На відміну від нативних програм, PWA працює як будь-який веб-сайт, тому його можна індексувати в пошукових системах. Порівняно зі звичайними веб-сторінками, PWA підвищила продуктивність та взаємодію, допомагаючи вашому веб-сайту зайняти краще місце в результатах пошуку[7].

Нижче наведено кілька додаткових кроків, щоб переконатися, що ваш PWA дружній до SEO:

Внесіть свою структуру SEO, подібну до вашого веб-сайту;

Використовуйте rel = canonical для кількох URL-адрес, щоб уникнути дублювання вмісту;

Зверніть увагу на URL-адреси зі знаком "#", оскільки Googlebot не буде індексувати нічого після символу.

Ви можете перевірити, чи Googlebot правильно сканує ваш сайт за допомогою таких інструментів, як Google Search Console.

Безпека: додаткові параметри безпеки в нативних програмах PWA є більш безпечними, ніж звичайні веб-програми, оскільки вони повинні працювати за протоколом HTTPS.

Ці протоколи безпеки гарантують, що жоден обмін інформації між клієнтом і сервером не підробляються. У захищеному середовищі клієнти можуть вводити особисті дані та дані кредитної картки, не турбуючись про їх крадіжку.

Порівняно з PWA, в нативнму додатку у вас є можливість вбудувати багато заходів безпеки. Якщо ваш додаток потребує входу, ви можете реалізувати багатофакторну автентифікацію. Ви також можете використовувати закріплення сертифіката для ще більш безпечного спілкування. Крім того, користувачі, швидше за все, більше довірятимуть програмі, аніж URL-адресу, оскільки перед публікацією вона повинна відповідати вимогам безпеки App Store або Google Store.

Завантаження та встановлення: PWA має просту установку без завантаження.

Середньостатистичний мобільний користувач встановлює нову програму приблизно на місяць. Певною мірою це пояснюється тим, що для досягнення завершення процесу встановлення та використання програми потрібен певний рівень прихильності. По-перше, користувачам потрібно знайти додаток у магазині та підтвердити, що вони хочуть його встановити. Після цього їм доведеться чекати завантаження та встановлення. Нарешті, після надання додатку певних дозволів, вони можуть користуватися програмою раз або двічі, перш ніж видалити її. Коли користувачі видаляють додаток, зазвичай це остаточне рішення, і вони можуть не повернутися до використання цього додатку.

З іншого боку, прогресивний веб-додаток не вимагає App Store або інсталяції. У веб-переглядачі відвідувачі можуть додати додаток на домашній екран за допомогою декількох натискань. PWA відображатиметься на головному екрані, у каталозі програм, надсилатиме сповіщення та інтегруватиметься в системні налаштування.

Крім того, прогресивний веб-додаток не займає стільки місця, скільки додаток. За допомогою лише URL-адреси відвідувачі можуть отримати доступ і поділитися програмою зі своїми друзями. Також немає необхідності оновлювати додаток, оскільки він завжди відображає останню версію під час запуску.

У таблиці 2.1 наведено вичерпний перелік функцій, доступних у PWA станом на січень 2021 року, а також їх сумісність. Зауваження слідують згодом.

Таблиця 2.1

Порівняння характеристик підходів

Особливість	Interpreted	PWA	Hybrid	Native
Встановлюється	Так	Така	Так	Так
Можливість роботи в автономному режимі	Так	Так	Так	Так
Тестування перед встановленням	Ні	Так	Ні	Ні
Доступність на ринку додатків	Так	Так(а)	Так	Так
Push-нотіфікацій	Так	Так(б)	Так	Так
Доступність між платформами	Так	Так	Так	Ні
Доступність API	Так	Обмежено(б)	Так(в)	Так
Фонова синхронізація	Так	Так	Так	Так

(а) PWA можуть бути встановленні як інтерпретовані додатки, що дозволяє їм стати повноцінними членами екосистеми. Підтримка push-повідомлень доступна через Push API [10], але обмежена деякими браузерами.

(б) PWA може використовувати API-інтерфейси на основі HTML5 для доступу до обладнання та платформі, до функцій які стали можливими завдяки Service Workers.

(в) Доступ до обладнання та платформи для гібридного додатка зазвичай надає Cordova, бібліотека для обробки мостів між компонентом веб-перегляду нативного додатка та API пристрою.

У таблиці 2.2 представлено порівняння трьох різних параметрів: розміру додатку, часу запуску та часу візуалізації. Розмір інсталяції прогресивного веб-додатка приблизно в 157 разів менший, ніж інтерпретований додаток на основі React Native, і приблизно в 43 рази менший, ніж гібридна програма на основі Ionic Framework. Якщо гібридна програма використовує більше 9 секунд для візуалізації панелі інструментів програми, інтерпретована програма використовує близько 860 мс для того самого завдання. PWA дала різні результати, коли (a) Chrome Canary не працював у фоновому режимі, і (б) він працював у фоновому режимі, незалежно від відкритого веб-сайту. Це пов'язано з залежністю браузера від PWA.

Таблиця 2.2

Вимірювання-порівняння підходів

Типи вимірювання	Hybrid	Interpreted	PWA
Розмір додатку	4.53MB	16.39MB	104KB
Час запуску	860ms	246ms	230ms
Час візуалізації	9242.1ms	862ms	(a) 3152ms
			(b) 1319ms

Порівняння продуктивності

У таблиці 2.2 представлені результати трьох вимірювань, проведених для попереднього збору даних. Тим самим він також служить опорою для подальших досліджень.

Результати ілюструють наявність певних компромісів. Якщо пріоритетом є мінімальний розмір програми, жоден підхід навіть віддалено не наближається до 104 КБ PWA. Другим найменшим за розміром є нативний додаток, який все ще важить у 42 рази більше (4,37 МБ), ніж PWA, хоча у деяких ситуаціях швидше відображається панель інструментів (із закритим Chrome). Найбільшим і найповільнішим із усіх підходів був крос-компільований додаток Xamarin. Цікаво, що крос-компільована програма працювала таким чином, оскільки база коду була

зібрана з загальної мови у рідні двійкові файли, відкриваючи таким чином власний інтерфейс користувача.

Тим не менш, гібридний підхід, який, як обговорювалося, викликає сумніви щодо продуктивності, зумів перевершити крос-компільований додаток у всіх трьох вимірах. Гібридний підхід також перевершив нативний підхід під час запуску активності (так само інтерпретується і PWA), але не зміг забезпечити ті ж швидкі швидкості візуалізації (нативний на 1688 мс, гібридний на 3999 мс).

Хоча розмір встановленого додатка в 157 разів перевищував PWA, він також у 3,5 рази швидше відтворював панель інструментів (перша візуалізація) під час запуску PWA без роботи Chrome у фоновому режимі. Якщо запустити з уже запущеним Chrome, розрив між двома програмами зменшився до 457 мс, що все ще надає перевагу інтерпретованому додатку. Якщо розмір установки не є основним пріоритетом проекту, інтерпретований підхід, здається, є продуктивним

Прогресивні веб-програми для єдиної розробки мобільних додатків, які швидко запускаються та швидко відтворюються, але займають місця у 1,5 рази менше, краще ніж програма з нативною компіляцією. Інтерпретовані та крос-компільовані підходи є єдиними крос-платформенними підходами, які б надавали нативний інтерфейс користувача. Якщо крос-платформенне розгортання та продуктивність нативного інтерфейсу є важливими критеріями проекту, на основі наших висновків буде застосований інтерпретований підхід.

Однак така робота зазвичай повідомляє про хороші результати від технологічно дуже обґрунтованих підходів, наприклад, від методів, заснованих на моделях, що керуються моделями, які були виключені в цій роботі, оскільки такі підходи не відіграють жодної практичної ролі в сучасній галузі розробки додатків.

Представлені результати обмежені тим, що програми тестувалися лише на одному конкретному пристрої, Galaxy S21, що працює під

управлінням Android 11.0. Проведені випробування не спираються на попередні дослідження або встановлені методи вимірювання продуктивності. Метою тестів було зібрати та представити попередні результати, щоб викликати інтерес для подальшої роботи. Відмінності можуть бути більш значними у старих, особливо менш потужних пристроях. Також можуть виникнути відмінності між технічними рамками підходу та програмами, які вони виробляють. Той факт, що фреймворк Xamarin із крос-компільованим підходом генерував такі результати порівняння з іншими підходами, не означає, що всі фреймворки з перехресною компіляцією будуть діяти однаково. Ми прагнемо проводити подальші дослідження щодо PWA та крос-платформенної розробки із залученням масиву пристроїв та фреймворків для підвищення валідності. Ми вважаємо, що наші висновки, представлені в таблиці 2.2, є важливими для прийняття рішень; тому продовжимо вивчати можливі виміри, технічні підходи та рамки, а також зміцнювати теорію, яка стоїть за роботою, як представлена тут.

Підхід «Прогресивна веб-програма» зараз очікує впроваджування Service Worker до веб-переглядача Apple. Від цього, ймовірно, залежить подальший розвиток цієї технолігії.

Якщо звернути увагу на швидкий час обробки інтерфейсу користувача (у деяких ситуаціях навіть швидше, ніж нативний додаток) [39], загальну добру сумісну інтеграцію платформи та пристрою через API HTML5 та JavaScript та незначний простір, який вони займають на пристрої, PWA слід вважати найкрашим майбутнім претендентом у крос-платформенному просторі.

У порівнянні з іншими кроссплатформенними підходами, PWA створює відсутність власних інтерфейсів користувача, крос-компільовані та інтерпретовані фреймворки. Однак, так само і популярний гібридний підхід. Доступ до API функцій пристрою та платформи обмежений у PWA, до яких такі API реалізуються у веб-

переглядачі користувачів. Для інших крос-платформенних підходів таких обмежень не виявлено. Однак, оскільки PWA на основі Інтернету можуть бути протестовані в будь-якому веб-браузері до можливої установки на пристрої у веб-переглядачі, сумісному з Service Worker. У порівнянні з іншими підходами це є унікальним для PWA. Співробітники служби надають службам PWA можливість виконувати такі завдання, як синхронізація у фоновому режимі (що дозволяє працювати в автономному режимі) та реєстрація push-сповіщень. Такі функції спочатку були доступні тільки в нативних програмах та за допомогою кроссплатформенних підходів.

Прогресивні веб-програми також мають потенціал стати рішенням для розробки настільних програм. З майбутніми планами Microsoft щодо включення PWA до свого магазину Windows, планами Google використовувати їх на своїй платформі ChromeOS та поточними можливостями встановлення PWA на настільних комп'ютерах, наприклад. у веб-переглядачі Chrome ми можемо побачити зміну в розробці настільних програм.

2.3 КОМПОНЕНТИ ПРОГРЕСИВНИХ ВЕБ -ДОДАТКІВ

Створення прогресивних веб-додатків та виконання всіх вимог, що ґрунтуються на продуктивності, доступності, найкращих практиках та SEO, є достатньо складним завданням. Щоб це сталося, усі компоненти PWA, наприклад, Service Worker, маніфест веб-додатків, модель оболонки додатків та сповіщення Web Push, повинні бути реалізовані з великою обережністю та працювати рука об руку. Усі ці компоненти описані в наступному розділі.

2.3.1 Маніфест веб-додатка

Маніфест веб-додатків - це простий файл JSON, що містить інформацію: ім’я, коротке_ім'я, опис, значки для різної роздільної здатності пристрою, режим відображення, колір теми додатку. Використання маніфесту веб-додатків встановлює веб-додаток на головному екрані користувача між нативними програмами. В результаті користувач може отримати швидкий доступ і насолоджуватися повноекранним досвідом, як у нативних програмі. Нижче наведено приклад файла manifest.json для веб-додатка.

```
{
  "short_name": "KSU24",
  "name": "KSU24",
  "icons": [
    {
      "src": "favicon.png",
      "sizes": "192x192",
      "type": "image/png"

}, {

      "src": "./splash_images/splash-512.png",
      "sizes": "512x512",
      "type": "image/png"

} ],

  "start_url": ".",
  "display": "standalone",
  "theme_color": "#000000",
  "background_color": "#ffffff",
  "gcm_sender_id": "482941778795"

}
```

Рисунок 2.1 - Приклад файла manifest.json для веб-додатка

Як показано вище, більшість термінів, що використовуються у файлі manifest.json, легко зрозуміти: name-це те, що з’являється на заставці, short_name це те, що з’являється на головному екрані, icons-

використовуються для зображення характеристики з масивом об’єктів, що складається з scr, розміру та типу. Подібним чином, start_url це URL -адреса, яка відкривається при натисканні на іконку; display контролює режим відображення, у якому запускається програма. Режим може бути автономним, повноекранним[20]. Автономний режим відкриває додаток без інтерфейсу веб-переглядача, наприклад, рядка розташування. Веб-додаток з файлом маніфесту надає додатку вигляд, як зображено на рисунку 2.2 [34].

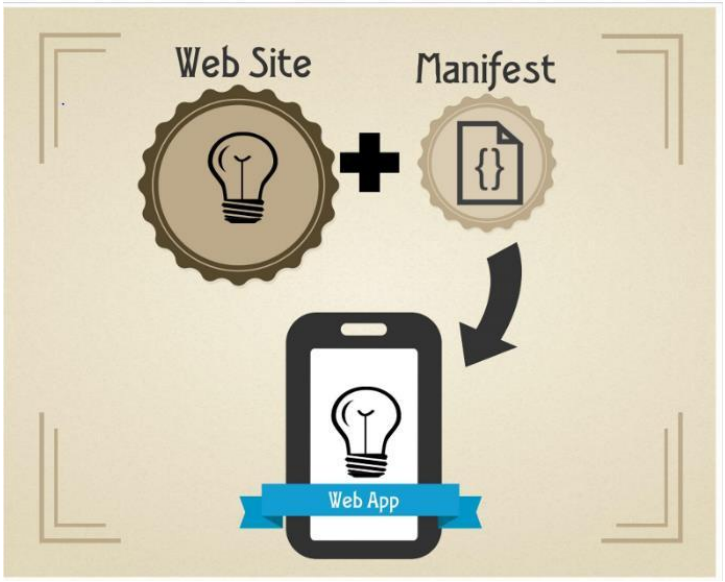


Рисунок 2.2 - Поєднання веб-сайту з маніфестом
для надання вигляду додатку.

Як показано на малюнку 2.2, веб-додаток поєднується з файлом маніфесту, так що користувач може встановити додаток на домашньому екрані свого пристрою. Маніфест веб-додатків пов'язаний з index.html сторінки програми, щоб веб-переглядачі могли ідентифікувати його, як показано нижче[21].

```
<head>
<link rel="manifest" href="/manifest.json">
</head>
```

Рисунок 2.3 - Приклад коду, який пов'язує index.htm з manifest.json

Щоб веб-додаток міг відображатися як банер для встановлення на веб-сайтах і забезпечувати подібність до програми, веб-додаток має відповідати таким критеріям:

1. Сайт повинен обслуговуватися через HTTPS.

2. На сайті має бути зареєстрований сервісний працівник.

3. Файл маніфесту веб-додатка на сайті повинен містити принаймні чотири обов’язкові назви полів або short_name (бажано обидва).

2.3.2 Впровадження технологій Service Worker

Сьогодні Веб-мережа використовується в практично в усіх аспектах нашого життя. Та коли при використанні користувачами веб-додатків виникають проблеми з підключенням до інтернету, а саме поганий зв'язок або втрата зв'язку, їм відображається сторінка "немає підключення до Інтернету", як показано на рисунку 2.4.

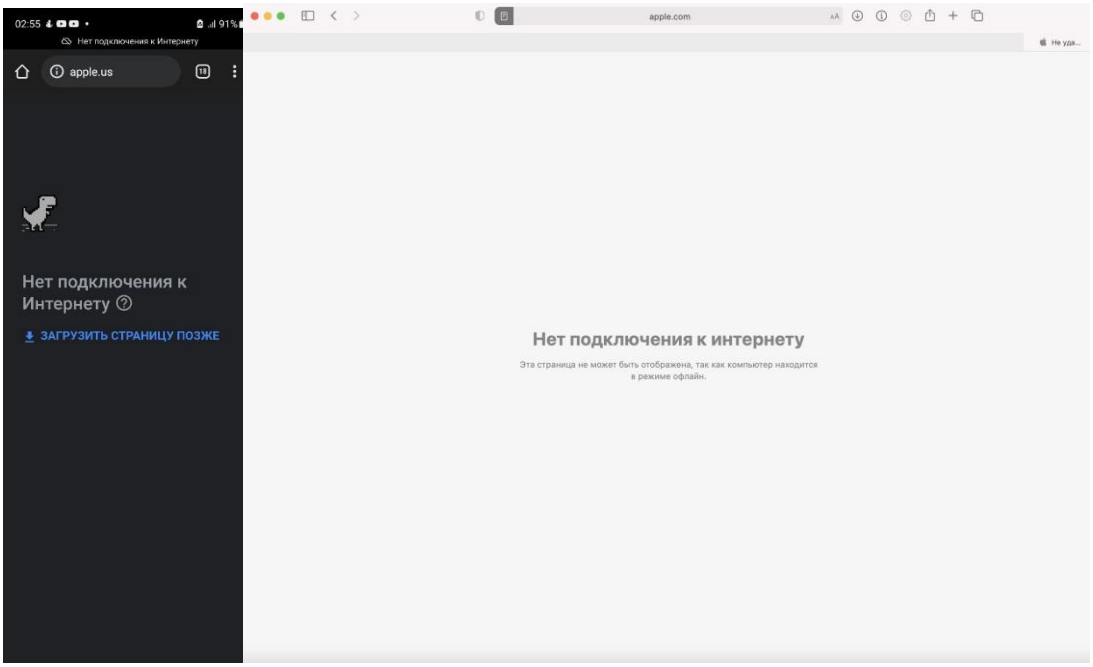


Рисунок 2.4 - Веб-додаток в автономному стані у режимі перегляду для мобільного та робочого столу

Як показано на рисунку 2.4, офлайн-стан веб-програми не може надавати корисної інформації користувачам. Але впровадження технологій Service Worker перетворило цю помилку на те, з чим можна злагоджено впоратися[22].

Service Worker дозволяє нам розширити функціональність нашого додатка певними функціями, доступними у нативних мобільних програмах. Приклади включають ввімкнення роботи в автономному режимі та push-сповіщення на екрані пристрою користувача нашої програми.

Технічно кажучи, Service Worker це свого роду проксі-сервер, розміщений між нашою програмою та Інтернетом. Завдяки цьому ми можемо кешувати елементи нашого веб-сайту (наприклад, графіку, HTML, файли CSS тощо), щоб використовувати їх під час роботи в автономному режимі та прискорювати роботу нашої програми (завантаження елементів із кешу, а не завантаження з Інтернету).

Service Worker реалізовано в окремому .js файлі (тому ми пишемо все за допомогою JavaScript) і пов'язаний з нашою програмою. Service Worker - це свого роду service worker, тому він запускається в окремому

потоці, абсолютно не пов’язаному з роботою програми, завдяки чому ми не блокуємо основний потік і не впливаємо (безпосередньо) на роботу нашої програми. Запуск в окремому потоці означає, що сервісний працівник не має доступу до елементів DOM сторінки, localStorage та AJAX. Варто також пам’ятати, що у файлі, в якому ми реалізуємо Service Worker, ми не можемо використовувати код «блокування». Отже, немає доступу до синхронного localStorage, тому нам доводиться покладатися на асинхронний код [40].

Service Worker, як я вже згадував раніше, не має доступу до запитів типу AJAX, тому він використовує інші переваги WEB API, такі як API Fetch, Cache API і лише асинхронні дані (обіцянки).

Сервісні працівники доступні лише за допомогою з'єднання HTTPS (за винятком localhost).

Останнє, що варто згадати, перш ніж приступати до реалізації, це те, що сервісний драйвер програми працює у фоновому режимі. Це означає, що він буде виконувати запрограмовані операції, навіть якщо додаток або браузер вимкнено. Завдяки цьому є можливість постійно зберігати елементи в кеші (для того, щоб використовувати їх, коли пристрій не в мережі) та надсилати push-сповіщення.

Service Worker - це досить потужний інструмент, який додає надзвичайно важливі функціональні можливості до нашого додатка. Серед них можна згадати роботу в автономному режимі, швидше завантаження ресурсів, обмеження використання підключення до Інтернету та push-сповіщення. Створення сервісного драйвера для програми можливо буде достатньо складним і кропітким завданням, особливо якщо програма високорозвинена [26]. На щастя, на допомогу приходять інструменти, які значно полегшують автоматичну генерацію файлу SW та його ефективну версію.

На мою думку, переваги використання Servive Worker настільки великі, що варто витратити час на його впровадження. Ми піднімемо

нашу програму на абсолютно новий рівень і наші користувачі неодмінно це оцінять.

2.3.3. Життєвий цикл Service Worker

Оскільки Service Worker є сценарієм, що керується скріптами, він має короткий термін служби. Він розпочинає роботу від скриптів і запускається, лише якщо йому потрібно обробити подію. За допомогою сервісного працівника розробник може належним чином контролювати кешування ресурсів [23]. Контроль над кешованими ресурсами відіграє важливу роль у розвитку офлайн-додаток, що є однією з ключових особливостей PWA. Через Service Worker веб-сторінка доступна навіть в автономному режимі, а кешовані дані завантажуються швидше навіть у нестабільній мережі. [13]

Service Worker має повністю відокремлений життєвий цикл від веб-сторінки. Рисунок 2.5 ілюструє спрощену версію життєвого циклу сервісного працівника при його першій установці.

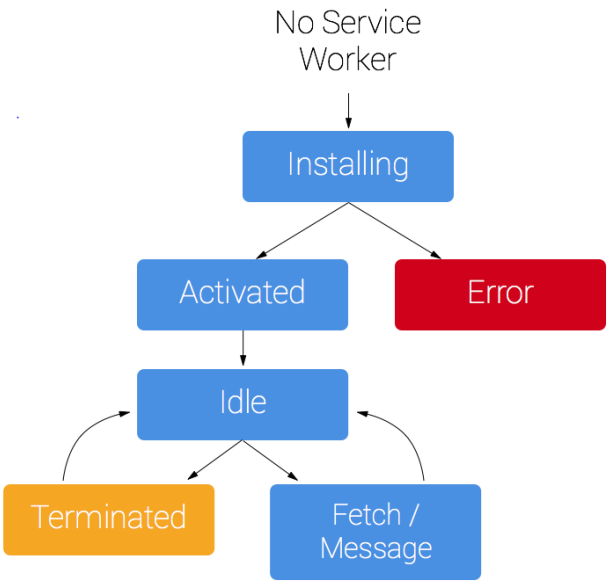


Рисунок 2.5 - Спрощена версія життєвого циклу Service Worker.

Як показано на рисунку 2.5, процес установки службового працівника починається з його реєстрації в JavaScript, де оголошується браузер, у якому зареєстрований файл JavaScript Service Worker[24]

```
if ('serviceWorker' in navigator) {
  window.addEventListener('load', function() {
    navigator.serviceWorker.register('/sw.js').then(function(registration) {
      // Registration was successful
      console.log('ServiceWorker registration successful with scope: ', regis-
tration.scope);
    }).catch(function(err) {
      // registration failed :(
      console.log('ServiceWorker registration failed: ', err);
    });
  });
}
```

Рисунок 2.6 - Перевірка служби підтримки веб-переглядача

Код рисунка 2.6 перевіряє, чи доступний API сервісного працівника, і якщо він є, службовий працівник у /sw.js реєструється під час завантаження сторінки.

Після завершення процесу реєстрації браузер починає встановлювати Service Worker, визначаючи зворотній дзвінок для події встановлення та налаштування середовища для кешування файлів.

Під час встановлення Service Worker кешує статичний вміст, а коли кешування проходить успішно, починається процес активації. Якщо процес кешування не вдається, то скрипт активації припиняється, і Service Worker спробує встановитись при наступному запуску додатку. Рисунок 2.7 ілюструє встановлення сервісного працівника.

```
var CACHE_NAME = 'my-site-cache-v1';
var urlsToCache = [
  '/',
  '/styles/main.css',
  '/script/main.js'
];

self.addEventListener('install', function(event) {
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then(function(cache) {
        console.log('Opened cache');
        return cache.addAll(urlsToCache);
      })
  );
});
```

Рисунок 2.7 - Встановлення Service Worker

Як показано на рисунку 2.7, успішний процес кешування та інсталяції запускає скрипт активації.

Активований Service Worker починає отримувати скрипти вибірки, натискання та синхронізації, коли користувач переходить на іншу сторінку.

2.3.4 Технології, використані для реалізації проекту

У цьому розділі описуються технології, використані для реалізації проекту у цьому дослідженні, такі як React.js та ES6, компоненти інтерфейсу матеріалу, Web Pack, Bable та Lighthouse. Інструменти, використані в проекті, були обрані з метою зменшення коду шаблону та полегшення процесу конфігурації.

React - це бібліотека JavaScript, яка використовується для створення інтерфейсу користувача. Facebook вперше представив React у 2013 році. Він був опублікований через два роки. React використовує віртуальний DOM для маніпулювання користувальницьким інтерфейсом, в результаті чого він забезпечує надзвичайно високу

продуктивність і робить сайт максимально інтерактивним, як показано на рисунку 2.8. [16,27]

Рисунок 2.8 - Візуалізація за допомогою VirtualDom у React.

Як показано на рисунку 2.8, React добре відстежує віртуальний DOM у пам'яті. Фейсбук, розробив React спеціально для вирішення однієї проблеми: створення великої програми з даними, які змінюються з плином часу. Таким чином, React прийняв декларативну стратегію на основі компонентів - навчись один раз і пиши будь-де. React спрощує створення інтерактивних інтерфейсів користувача. Розробляючи просте спостереження для кожного стану програми, React плавно оновлює та відтворює лише необхідні компоненти при зміні даних. Декларативний перегляд в React робить код легким для передбачення та налагодження.

React використовує структуру на основі компонентів. Логіка компонентів написана в JavaScript замість шаблонів, що полегшує проходження багатьох даних через веб-додаток та утримує стан поза DOM. Проблеми вирішуються шляхом створення компонентів багаторазового використання, а коли компоненти ускладнюються, вони розбиваються на менші та простіші. Компоненти React подібні до функції JavaScript. [27]

React також використовується на стороні сервера за допомогою NodeJs та потужних мобільних додатків за допомогою React Native. Усі ці функції разом роблять React найулюбленішем фреймворком серед розробників порівняно з іншими фреймворками та бібліотеками. Він випередив своїх конкурентів AngularJs, Angular 2 та Ember протягом

трьох років з моменту його випуску, як показано на рисунку 2.9. Тепер React прийнято за парадигму у світі інтернет-розробки.

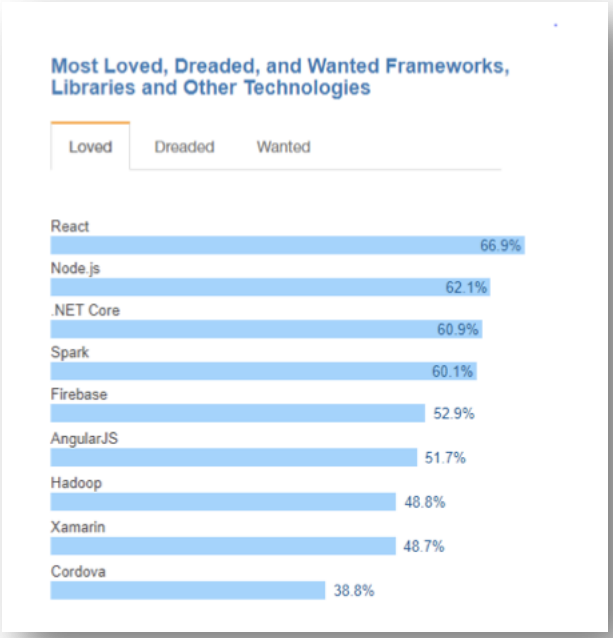


Рисунок 2.9 - Найулюбленіші технології 2021 року у веб-розробці

Як показано на рисунку 2.9, React була найулюбленішою технологією у 2021 році - 66,9%, тоді як найменш улюбленим фреймворком є Cordova - лише 38,8% [28].

React використовує XML-подібний синтаксис JSX, розширення синтаксису до ECMAScript для опису представлення DOM компонента. Він схожий на HTML або XML мови розмітки, але орієнтований на JavaScript. Це робить код більш читабельним, а написання JSX створює відчуття написання HTML. Вона базується на поділі проблем, а не на технології, оскільки поєднує розмітку, стиль та поведінку компонентів в одному файлі, а не в кожному окремому файлі. Однак використання JSX у додатку React не є обов'язковим, але використання JSX допомагає спростити розробку, обробку помилок та підвищити продуктивність програми React.

React розподілен на компоненти, вони схожі на функції JavaScript, які дозволяють розділити інтерфейс користувача на незалежні частини, які можна використовувати багаторазово. Компоненти є будівельними матеріалами в додатку React. Компоненти React створюються за допомогою методу React.createClass (). Компоненти приймають довільні вхідні дані, які називаються реквізитами та станом. Props - це канал зв'язку, який діє як міст між батьківськими та дочірніми компонентами. Реквізит використовується для відображення статичних незмінних даних, тоді як тип даних стану використовується для динамічних даних.

РОЗДІЛ 3
РОЗРОБКА ПРОГРЕСИВНОГО ВЕБ-ДОДАТКУ

3.1 Створеня React додатку KSU24

3.1.1 Налаштування React

Налаштування React - це інструмент для початку роботи з середовищем розробки програми React.js. Створення програми реагування за допомогою попередньо налаштованих Webpack, Bable та інших необхідних інструментів робить процес розробки програми плавним і без проблем. Він також поставляється з попередньо налаштованим сервісним працівником, який є одним із ключових компонентів PWA. Додаток Create React можна легко встановити з терміналу за допомогою команди: npm install -g create-React-app, як показано на додатку [В].

Як показано на додатку [В], програма **create-React-pwanews** створює нову програму React у ~/pwanews. Зміна каталогу на pwanews і запуск **npm start** запускає сценарій реакції, що запускає сервер.

Як показано на додатку [Г], після успішної компіляції програми create-React-app її можна переглянути у браузері, відвідавши http: // localhost: 3000 [29].

3.1.2 Webpack

Webpack - це пакет статичних модулів для сучасних програм JavaScript. Сучасні програми JavaS-cript мають різні модулі та таблиці стилів. Ці модулі та таблиці стилів спрощують процес розробки. Однак під час розгортання це створює клопоти. Коли використовується веб-пакет, він обробляє додаток і рекурсивно будує графік залежності, який

включає кожен модуль, необхідний програмі, і упаковує всі ці модулі в один або кілька пакетів. Webpack покращує продуктивність програми. [30]

3.1.3 Babel

Babel - це компілятор JavaScript. Він перетворює нову версію коду JavaScript на стару версію. У поєднанні з Webpack Babel переносить ES6 та JSX на ES5. Це зроблено для підтримки максимальної кількості можливих користувачів [31].

3.1.4 Інші онлайн-сервіси

RSS в JSON - це онлайн-сервіс для перетворення RSS-каналу в REST API, який потім може використовуватися будь-якими програмами, які підтримують отримання даних з даного API. Додаток KSU24, розроблений у проекті, отримує свої дані від цього сервісу [43].

3.1.5 Cloudiary

Cloudiary - це платформа для управління медіа для веб-та мобільних розробників. Cloudinary використовується як платформа для оптимізації зображення та відео. Cloudinary зіграв важливу роль у підвищенні продуктивності PWA шляхом оптимізації розміру зображення. Один сигнал і Zapier використовуються для реалізації push-сповіщення [12].

3.2 Опис використаних апаратних та програмних ресурів

У цьому розділі описується реалізація прототипу додатка KSU24, розробленого в цьому дослідженні для демонстрації особливостей та

переваг PWA. Розроблена таким чином програма містить усі компоненти PWA: Service Worker, Web Manifest, Shell App та Web push-сповіщення. За допомогою додатку KSU24 користувач може переглядати новини університету, а також отримувати сповіщення про останні новини за допомогою push-сповіщень, встановлювати додаток на головному екрані одним натисканням та використовувати офлайн з інтерфейсом користувача та інтерфейсом користувача, як нативний додаток.

Додаток було розроблено на MacBookPro з апаратним забезпеченням операційної системи macOS Big Surr версії 11.2.1. Node.js та додаток для створення React були завантажені та об’єднані для створення середовища для додатку KSU24. Крім цього, у проекті також були використані наступні інструменти:

Visual Studio Code - це безкоштовний, легкий і надійний редактор коду, який допомагає в розробці додатків за допомогою таких функцій, як вбудований CLI та вбудована підтримка git. Він доступний для всіх платформ: macOS. Windows та Linux. Він має інтелектуальне завершення коду, спрощене налагодження та контроль джерела в продукті. Він також має вбудовану підтримку Typescript, JavaScript та Node.js та розширення для C#, C ++, Java, Python, PHP та Go. Крім того, він також має середовища, такі як Unity та .NET.’

NPM використовується для менеджера пакетів. Це швидкий, надійний і безпечний менеджер пакетів. Так само Firebase використовувався для розміщення проекту. Firebase забезпечує з'єднання HTTPS, тоді як Git використовувався для контролю версій [35].

3.3 Реалізація прогресивного веб-додатку KSU24

Фаза реалізації проекту розпочалася з інсталяції Node.js, а потім-create- React-app. Babel та Webpack були попередньо налаштовані у

програмі Create React. Так само були імпортовані матеріальні компоненти інтерфейсу користувача.

Процес застосування цього проекту досить простий: отримання стрічки новин RSS з головного сайту університета та відображення результату. На рисунку 3.1 зображена структура заявок на повний проект PWA.

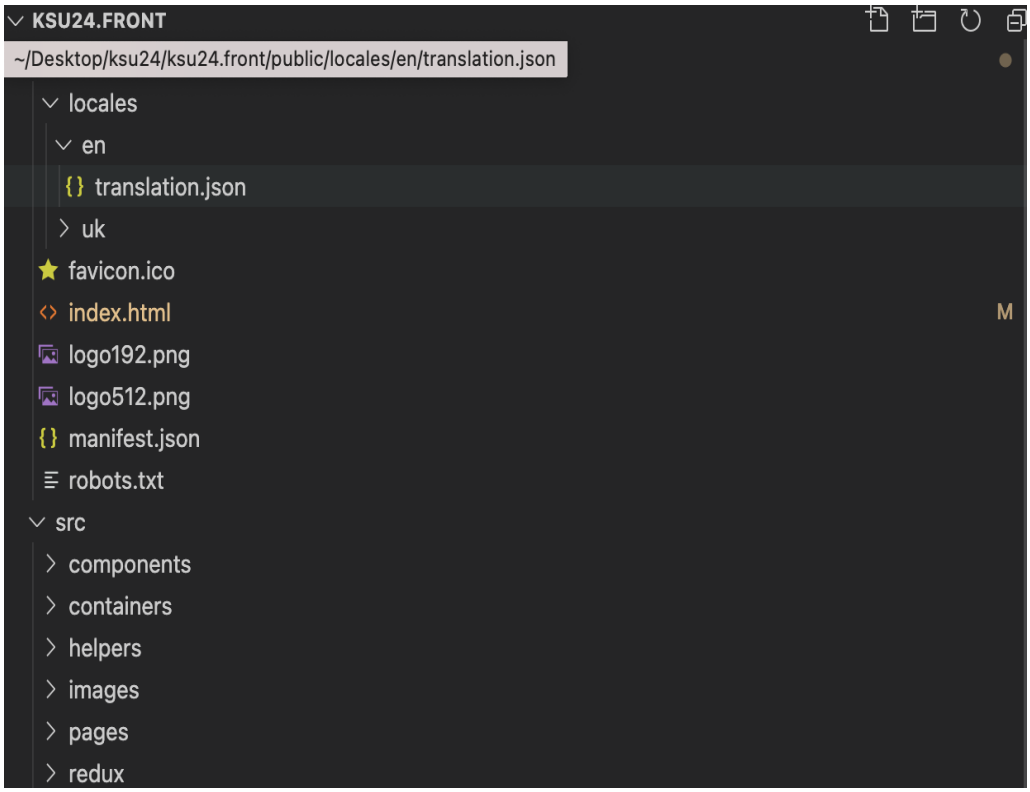


Рисунок 3.1 - Структура заявок на повний проект PWA

Як показано на рисунку 3.1, структура папок програми містить усі файли, необхідні для розробки програми. Усі файли були включені до батьківського каталогу KSU24.

Відкритий каталог містить усі ресурси, які обслуговуються безпосередньо, без додаткової обробки веб-пакетом. Він складається з index.html, manifest.json, значків і файлів для OneSignal. Index.html діє як точка входу для веб-програми. Файл Manifest.json містить всю конфігурацію щодо того, як поводиться PWA при додаванні на головний екран.

Каталог src є координатором програми create React. Він містить усі файли Javascript, які згодом обробляються webpack. Він складається з активів, компонентів, контейнерів та корисних функцій. Uitls складається з App.js, AsyncComponents.js, index.css, index.js, registerServiceWorker.js та Routes.js. App.js є основним компонентом JavaScript, тоді як index.js просто вставляє програму в документ. RegisterServiceWorker.js асоціюється з кешуванням та оновленням файлів для кінцевого користувача і відповідає за кешування в автономному режимі та швидше завантаження сторінки після першого відвідування користувача програмою.

Файл "package.json" містить всю інформацію щодо програми та її залежностей, які використовуються для створення та запуску програми.

На додатку Д наведено файл „package.json”, який містить детальну інформацію про ім’я, версію, сценарії та всі залежності, які були використані у проекті [33].

Node_modules містить усі пакети, але для цього проекту використовуються реакція та інші залежності, зазначені у файлі package.json.

Gitignore - це стандартний файл, який використовується контролем версій git для вирішення, які файли та каталоги ігнорувати.

Після успішного завершення конфігурації, налаштування, кодування, тестування та налагодження проект був успішно виконаний. В результаті було розроблено додаток KSU24.

Розроблений таким чином прототип додатку KSU24 дотримувався всіх вказівок і пройшов тест, проведений за допомогою інструменту Lighthouse. Lighthouse - це автоматизований інструмент з відкритим кодом, який використовується для аудиту веб-сторінок з точки зору доступності, прогресивного веб-додатка, продуктивності та SEO. Він може працювати різними способами як розширення для Chrome, або з командного рядка, або як модуль Node. Lighthouse проводить серію

перевірок щодо сторінки вказаної URL-адреси та формує звіт про те, хто цю сторінку виконував. В результаті він також представляє невдалі аудити як показник того, як можна покращити оцінку сторінки. Кожен аудит має довідковий документ, який пояснює необхідність аудиту та способи його виправлення.

PWA розміщено по URL: https://ksu24.kspu.edu URL-адреса програми новин PWA була перевірена Lighthouse. В додатку Е ілюструє звіт оцінки Lighthouse для програми KSU24

Як показано в додатку Е, оцінка Lighthouse показала, що додаток на 100 відсотків прогресивний і відповідає всім вимогам бути прогресивним веб-додатком, з 85-процентним балом у продуктивності, 81 у доступності, 88 у найкращих практиках та 78 у SEO. Прототип перевірено на реальному пристрої [41]. Додаток KSU24 у режимі перегляду для мобільних пристроїв зображено на додатку Є.

Розроблений таким чином додаток KSU24 може працювати в автономному режимі, встановлюватись на домашньому екрані, запускатись у повноекранному режимі, надсилати Push-повідомлення, і швидко завантажуватись навіть у нестабільних мережах.

ВИСНОВКИ

Основною метою магістерскої роботи було оцінити та впровадити PWA до інфраструктури університета. Значний час було витрачено на вивчення теми та її складових. Плюси та мінуси були проаналізовані шляхом порівняння PWA з іншими технологіями. PWA не є заміною нативних програм. Це лише новий підхід до того, щоб зробити веб-програми більш схожими на нативні. Він швидко набирає популярність завдяки своїм функціям, більш простому процесу розробки, створенню та впровадженню будь-якої стратегії. Однак PWA стикається з різними проблемами, наприклад з підтримкою браузера для різних функціональних можливостей. Firefox та Safari мають деякі проблеми з маніфестом додатків та Service Worker. Маніфест веб-додатків знаходиться у стадії розробки у Firefox, і нещодавно Safari розпочав підтримку Service Worker у своїй останній версії 11.1. Однак Chrome, Chrome для Android, UC Browser для Android та Samsung Internet мають хорошу підтримку всіх функцій.

Після детального вивчення відповідної теорії та архітектури веб-додатків акцент було зміщено на практичну реалізацію. Таким чином, був розроблений "PWA KSU24", який ілюстрував впровадження Progressive Web App та його компонентів до інфраструктури університета. Розроблений таким чином PWA виглядає та використовується як нативний додаток. Таким чином, данний проект був успішно розроблений та впровадженний до інфраструктури університета. Всі цілі, які були поставлені в даній роботі повністю реалізовані.

На підставі досліджень та практичної реалізації можливо зробити висновок, що PWA може виглядати як модне слово навколо покращеного веб-стандарту, але позитивний вплив, який він приніс як на підприємства, так і для користувачів, незаперечний. Він приніс функції

та вдосконалення для мобільного Інтернету, які до цього часу були виключно привілеї нативних програм. Навіть такі компанії, як AliExpress, Housing.com, Twitter Lite, Flipkart, Forbes, Financial Times, Google I/O, і BookMyShow впровадили PWA, завдяки чому відбулося розширення їх бізнесу. Крім того, Google докладає значних зусиль для надання навчальних посібників та підтримки PWA. Таким чином, завдяки більшій підтримці браузерів і платформ, PWA може стати майбутнім мобільного Інтернету.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Review of faramework terminology [Електронний ресурс]. – Режим доступу: URL: https://riehle.org/computer-science/research/dissertation/chapter-2.html
2. Співаковський О. В. Функції та структура університету як складного механізму, який обслуговує освітні інтереси [Електронний ресурс] / Олександр Володимирович Співаковський // 12. – 2012. – Режим доступу до ресурсу: http://nbuv.gov.ua/UJRN/itvo_2012_12_5
3. Фреймворки у веб-розробці. [Електронний ресурс]. – Режим доступу: URL: https://web-creator.ru/articles/about_frameworks.
4. Rakesh Vidya Chandra, Bala Subrahmanyam Varanasi (2015). Python Requests Essentials. ISBN 978-1-78439-541-4
5. Чому використовувати Django [Електронний ресурс]. – Режим доступу: URL: https://djangostars.com/blog/why-we-use-django-framework
6. Офіційна документація Django (веб-фреймворк) [Електронний ресурс]. – Режим доступу: URL: https://docs.djangoproject.com/en/dev/faq/general/
7. How to make PWA [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.codica.com/blog/how-to-create-pwa-with-react/.
8. Dawning of Progressive Web Applications (PWA): Edging Out the Pitfalls of Traditional Mobile Development [Електронний ресурс] // Vol68 No1. – 2020. – Режим доступу до ресурсу: https://asrjetsjournal.org/index.php/American_Scientific_Journal/article/view/5812.
9. Офіційна документація Flask [Електронний ресурс]. – Режим доступу: URL: http://flask.pocoo.org/docs/1.0/foreword/#what-does-micro-mean
10. Push API [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/API/Push_API.

11. Aslam, M., Auer, S., Shen, J. and Herrmman, M. Expressing Business Process Models as OWL-S Ontologies. Workshop on Grid and Peer-to-Peer Based Workflows (GPWW) in conjunction with BPM 2006. LNCS 4103, pp. 400-415 (2006)

12. Cloudiary [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://imageboss.me/docs.

13. Miguel Grinderg (2018). Flask Web Development. ISBN 978-1-491-99173-2

14. Документація додатку «pip» [Електронний ресурс]. – Режим доступу: URL: https://pypi.org/project/pip/

15. Norton, B., Cabral, L. and Nitzsche, J. Ontology-based Translation of Business Process Model. The Fourth International Conference on Internet and Web Applications and Services (ICIW 2009), Venice, Italy, IEEE Computer Society. (2009)

16. Офіційна документація шаблонізатора Jinja2 [Електронний ресурс]. – Режим доступу: URL: http://jinja.pocoo.org/

17. Ouyang, C., Aalst, W., Dumas, M. and Hofstede, A.: From Business Process Models to Process-oriented Software Systems: The BPMN to BPEL Way [Електронний ресурс]. – Режим доступу: URL: http://eprints.qut.edu.au/ archive/00005266/01/5266.pdf

18. Recker, J. and Mendling, J. On the Translation between BPMN and BPEL: Conceptual Mismatch between Process Modeling Languages [Електронний ресурс]. – Режим доступу: URL: http://eprints.qut.edu.au/ archive/00004637/01/4637.pdf

19. What is PostgreSQL? *PostgreSQL Documentation*. URL: https://www.postgresql.org/docs/11/intro-whatis.html

20. Web app manifest [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://web.dev/add-manifest/.

21. Authentication. *NestJS Documentation*. URL: https://docs.nestjs.com/techniques/authentication

22. OAuth2orize: authorization server toolkit for Node.js. Docs. URL: https://github.com/jaredhanson/oauth2orize

23. Middleware. *NestJS Documentation.* URL: https://docs.nestjs.com/middleware

24. NestJS Admin Docs. URL: https://nestjs-admin.com/

25. Sivashanmugam, K., Miller, J., Sheth, A. and Verma K. Framework for Semantic Web Process Composition. International Journal of Electronic Commerce, Winter 2004– 5, Vol. 9, No. 2, pp. 71–106 (2005)

26. Campbell B., Mortimore C., Jones M. Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants. *IETF-related tools.* 2015. RFC 7522. URL: https://tools.ietf.org/html/rfc7522

27. Czym jest React i jakie ma zalety? [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://global4net.com/ecommerce/czym-jest-react-i-jakie-ma-zalety/.

28. Kaler C., McIntosh M. Web Services Federation Language (WS-Federation) Version 1.2. *OASIS Standard*. 2009. URL: http://docs.oasis-open.org/wsfed/federation/v1.2/os/ws-federation-1.2-spec-os.html

29. Install React directly [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows/dev-environment/javascript/react-on-windows.

30. Bolesław N. Czym jest Webpack i kiedy warto go używać? [Електронний ресурс] / Nius Bolesław. – 2019. – Режим доступу до ресурсу: https://global4net.com/ecommerce/czym-jest-webpack-kiedy-uzywac/.

31. Dawid R. Co to jest Babel? [Електронний ресурс] / Dawid – Режим доступу до ресурсу: Bolesław N. Czym jest Webpack i kiedy warto go używać? [Електронний ресурс] / Nius Bolesław. – 2019. – Режим доступу до ресурсу: https://global4net.com/ecommerce/czym-jest-webpack-kiedy-uzywac/.

32. NestJS Documentation. URL: https://docs.nestjs.com/

33. What is the file `package.json`? 2011. URL: https://nodejs.org/en/knowledge/getting-started/npm/what-is-the-file-package-json/

34. Co to jest Service Worker.js [Електронний ресурс] // spaceout. – 2020. – Режим доступу до ресурсу: https://blog.spaceout.pl/co-to-jest-service-worker-js-aplikacja-progresywna-od-google/.

35. Code editing. Redefined. [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://code.visualstudio.com.

36. Barros, O. Business Process Patterns and Frameworks: Reusing Knowledge in Process Innovation, Business Process Management Journal, January 2007

37. Abu Naser S. S., Mahmoud Abu Ghosh , and Rasha R. Atallah, " Using Social network in Higher Education A case Study on the University of Palestine" , IJERA, 2014.

38. Rasha R. Atallah, Abu Naser S. S., "Data Mining Techniques in Higher Education an Empirical Study for the University of Palestine", IJMER, 4(4) ,48-52, 2014.

39. Abu Naser S. S., Rasha R. Atallah, Sahar Hamo, "Building an Ontology in Educational Domain Case Study for the University of Palestine", International Journal of Research in Engineering and Science (IJRES), 3(1), 2015.

40. Co to jest Service Worker.js [Електронний ресурс] // spaceout. – 2020. – Режим доступу до ресурсу: https://blog.spaceout.pl/co-to-jest-service-worker-js-aplikacja-progresywna-od-google/.

41. Lighthouse PWA Analysis Tool [Електронний ресурс] // google. – 2021. – Режим доступу до ресурсу: https://developers.google.com/web/ilt/pwa/lighthouse-pwa-analysis-tool.

42. Barros, O. A Novel Approach to Joint Business and Information System Design, Journal of Computer InformationSystems,XLV, 3,p96.Spring2005.

43. Co to jest RSS [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://sprawnymarketing.pl/czym-jest-rss-feed/.

44. Сенчишен Д. О. Проєктування та розроблення сервісної архітектури управління бізнес-процесами університету. Сервіс «Відділ кадрів» [Електронний ресурс] / Д. О. Сенчишен. – 2020. – Режим доступу до ресурсу: http://ekhsuir.kspu.edu/handle/123456789/13482.

45. Лукінов Г. Г. Використання реактивного програмування при створенні Web UI [Електронний ресурс] / Лукінов Г. Г. – Режим доступу до ресурсу: http://ekhsuir.kspu.edu/handle/123456789/11899

46. Романенко, В. М. РОЗРОБЛЕННЯ СИСТЕМИ УПРАВЛІННЯ ОНЛАЙН КОНФЕРЕНЦІЯМИ BIGBLUEBUTTON ДЛЯ ІНТЕГРАЦІЇ З LCMS MOODLE KSUONLINE [Електронний ресурс] / Романенко, В. М.. – 2021. – Режим доступу до ресурсу: http://ekhsuir.kspu.edu/handle/123456789/14243.

47. РОЗРОБЛЕННЯ МОБІЛЬНОГО НАВЧАЛЬНОГО ДОДАТКУ ДЛЯ ПІДГОТОВКИ ДО ВСТУПУ НА НАВЧАННЯ НА ДРУГОМУ (МАГІСТЕРСЬКОМУ) РІВНІ ВИЩОЇ ОСВІТИ [Електронний ресурс] // 2021 – Режим доступу до ресурсу: http://ekhsuir.kspu.edu/handle/123456789/14255.

ДОДАТКИ

Додаток А.

КОДЕКС АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ ХЕРСОНСЬКОГО ДЕРЖАВНОГО УНІВЕРСИТЕТУ

Я, Шкворець Владислав Владленович , учасник(ця) освітнього процесу Херсонського державного університету, **УСВІДОМЛЮЮ**, що академічна доброчесність – це фундаментальна етична цінність усієї академічної спільноти світу.

ЗАЯВЛЯЮ, що у своїй освітній і науковій діяльності **ЗОБОВ'ЯЗУЮСЯ**:

– дотримуватися:

- вимог законодавства України та внутрішніх нормативних документів університету, зокрема Статуту Університету;
- принципів та правил академічної доброчесності;
- нульової толерантності до академічного плагіату;
- моральних норм та правил етичної поведінки;
- толерантного ставлення до інших;
- дотримуватися високого рівня культури спілкування;

– надавати згоду на:

- безпосередню перевірку курсових, кваліфікаційних робіт тощо на ознаки наявності академічного плагіату за допомогою спеціалізованих програмних продуктів;
- оброблення, збереження й розміщення кваліфікаційних робіт у відкритому доступі в інституційному репозитарії;
- використання робіт для перевірки на ознаки наявності академічного плагіату в інших роботах виключно з метою виявлення можливих ознак академічного плагіату;

– самостійно виконувати навчальні завдання, завдання поточного й підсумкового контролю результатів навчання;

– надавати достовірну інформацію щодо результатів власної навчальної (наукової, творчої) діяльності, використаних методик досліджень та джерел інформації;

– не використовувати результати досліджень інших авторів без використання покликань на їхню роботу;

– своєю діяльністю сприяти збереженню та примноженню традицій університету, формуванню його позитивного іміджу;

– не чинити правопорушень і не сприяти їхньому скоєнню іншими особами;

– підтримувати атмосферу довіри, взаємної відповідальності та співпраці в освітньому середовищі;

– поважати честь, гідність та особисту недоторканність особи, незважаючи на її стать, вік, матеріальний стан, соціальне становище, расову належність, релігійні й політичні переконання;

– не дискримінувати людей на підставі академічного статусу, а також за національною, расовою, статевою чи іншою належністю;

– відповідально ставитися до своїх обов'язків, вчасно та сумлінно виконувати необхідні навчальні та науково-дослідницькі завдання;

– запобігати виникненню у своїй діяльності конфлікту інтересів, зокрема не використовувати службових і родинних зв'язків з метою отримання нечесної переваги в навчальній, науковій і трудовій діяльності;

– не брати участі в будь-якій діяльності, пов'язаній із обманом, нечесністю, списуванням, фабрикацією;

– не підроблювати документи;

– не поширювати неправдиву та компрометуючу інформацію про інших здобувачів вищої освіти, викладачів і співробітників;

– не отримувати і не пропонувати винагород за несправедливе отримання будь-яких переваг або здійснення впливу на зміну отриманої академічної оцінки;

– не залякувати й не проявляти агресії та насильства проти інших, сексуальні домагання;

– не завдавати шкоди матеріальним цінностям, матеріально-технічній базі університету та особистій власності інших студентів та/або працівників;

– не використовувати без дозволу ректорату (деканату) символіки університету в заходах, не пов'язаних з діяльністю університету;

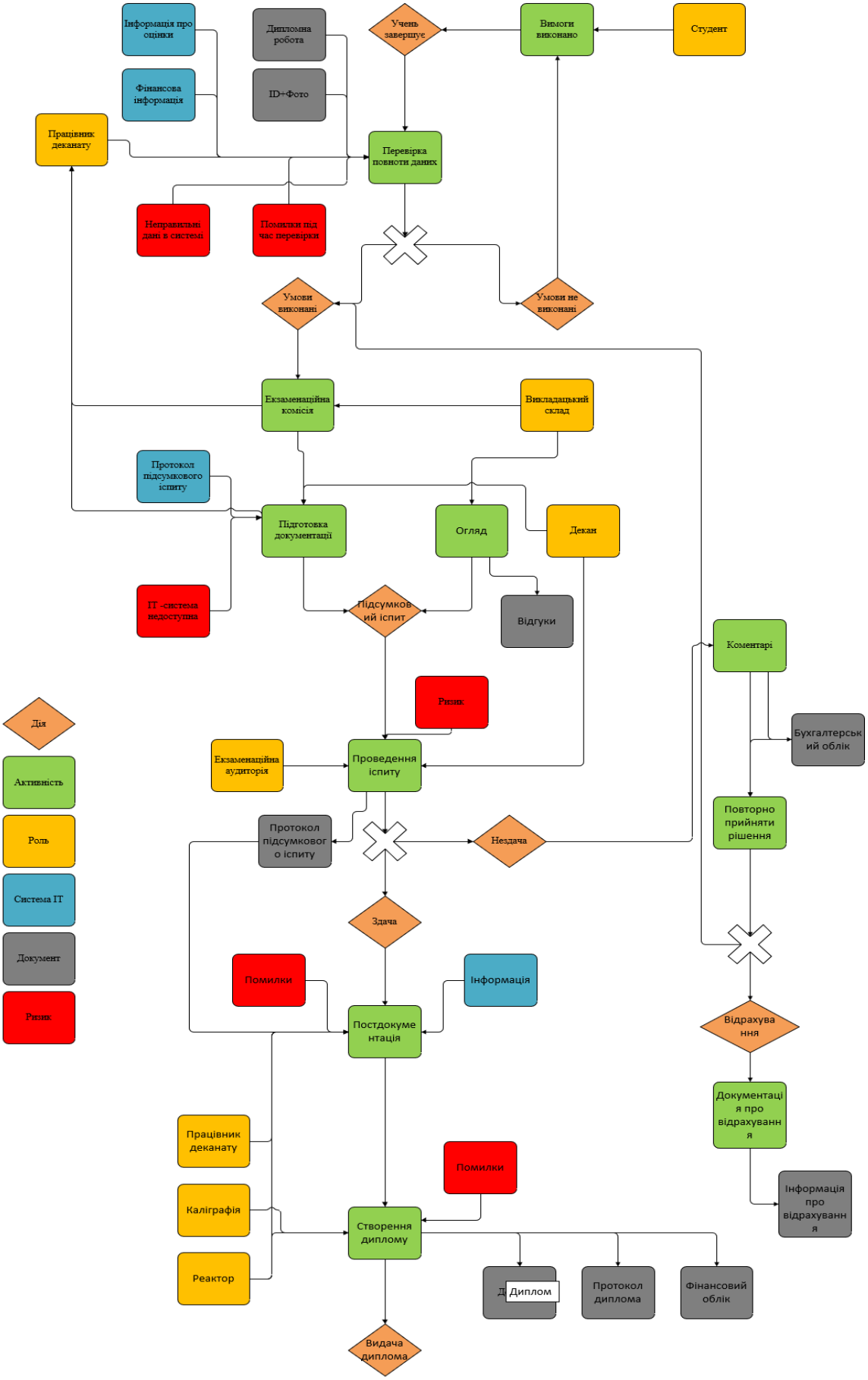
– не здійснювати і не заохочувати будь-яких спроб, спрямованих на те, щоб за допомогою нечесних і негідних методів досягати власних корисних цілей;

– не завдавати загрози власному здоров'ю або безпеці іншим студентам та/або працівникам.

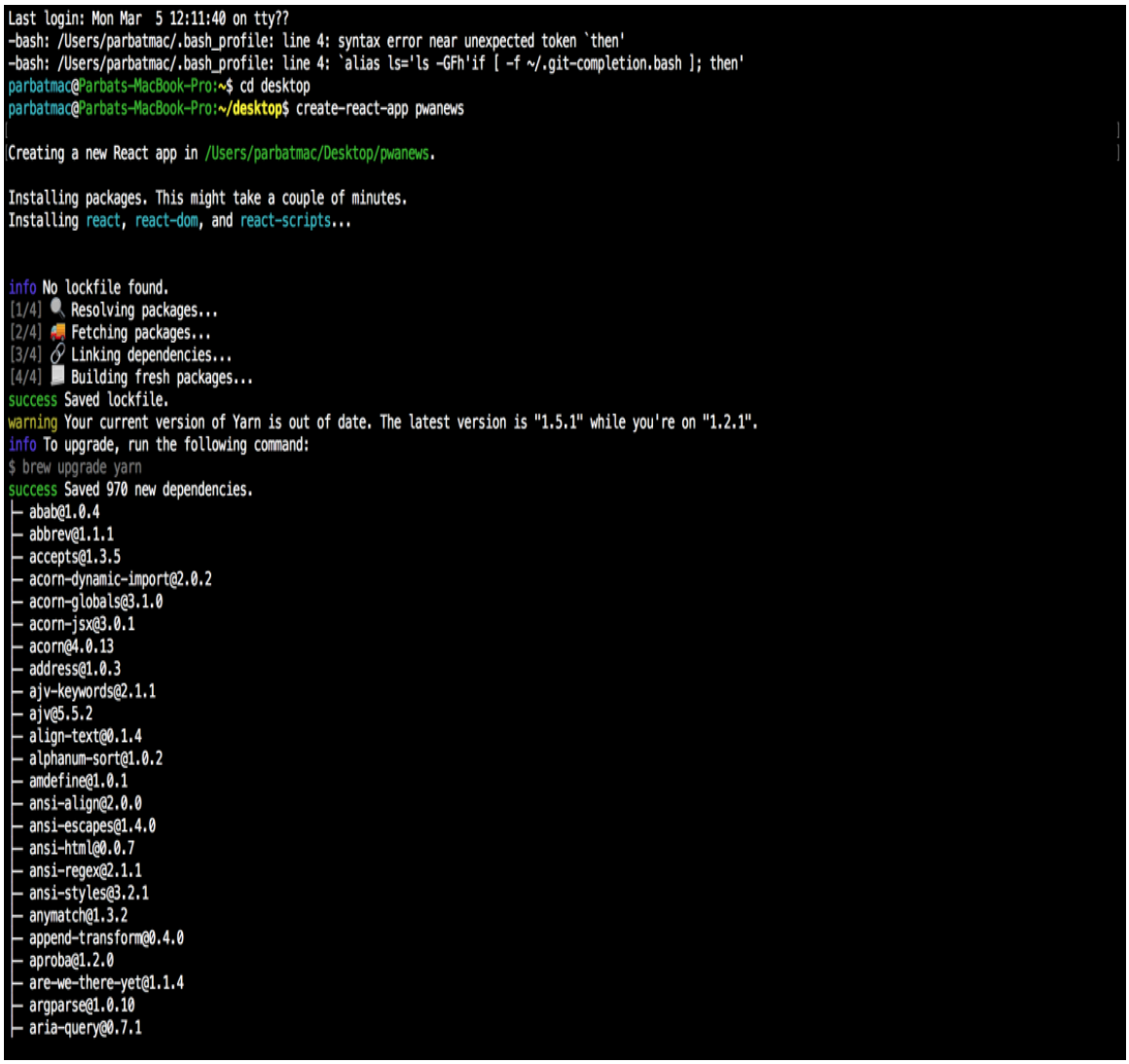
УСВІДОМЛЮЮ, що відповідно до чинного законодавства у разі недотримання Кодексу академічної доброчесності буду нести академічну та/або інші види відповідальності й до мене можуть бути застосовані заходи дисциплінарного характеру за порушення принципів академічної доброчесності.

07.09.2020		Шкворець В.В.
(дата)	(підпис)	(ім'я, прізвище)

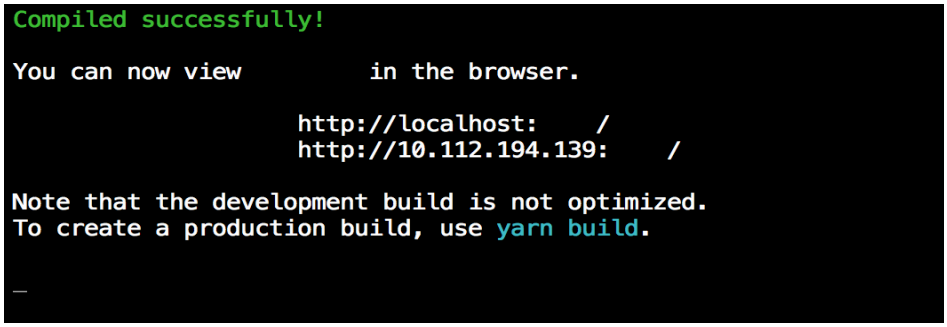
Додаток Б.


Інформація про оцінки
Дипломна робота
Учень завершує
Вимоги виконано
Студент
Фінансова інформація
ID+Фото
Працівник деканату
Перевірка повноти даних
Неправильні дані в системі
Помилки під час перевірки
Умови виконані
Умови не виконані
Екзаменаційна комісія
Викладацький склад
Протокол підсумкового іспиту
Підготовка документації
Огляд
Декан
IT -система недоступна
Підсумковий іспит
Відгуки
Коментарі
Ризик
Бухгалтерський облік
Екзаменаційна аудиторія
Проведення іспиту
Повторно прийняти рішення
Протокол підсумкового іспиту
Нездача
Здача
Помилки
Інформація
Постдокументація
Відрахування
Документація про відрахування
Працівник деканату
Помилки
Інформація про відрахування
Каліграфія
Створення диплому
Реактор
Диплом
Протокол диплома
Фінансовий облік
Видача диплома
Дія
Активність
Роль
Система IT
Документ
Ризик

Додаток В.


Last login: Mon Mar 5 12:11:40 on tty??
-bash: /Users/parbatmac/.bash_profile: line 4: syntax error near unexpected token `then'
-bash: /Users/parbatmac/.bash_profile: line 4: `alias ls='ls -GFh'if [-f ~/.git-completion.bash]; then'
parbatmac@Parbats-MacBook-Pro:~$ cd desktop
parbatmac@Parbats-MacBook-Pro:~/desktop$ create-react-app pwanews
Creating a new React app in /Users/parbatmac/Desktop/pwanews.
Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts...
info No lockfile found.
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
[4/4] Building fresh packages...
success Saved lockfile.
warning Your current version of Yarn is out of date. The latest version is "1.5.1" while you're on "1.2.1".
info To upgrade, run the following command:
$ brew upgrade yarn
success Saved 970 new dependencies.
├─ abab@1.0.4
├─ abbrev@1.1.1
├─ accepts@1.3.5
├─ acorn-dynamic-import@2.0.2
├─ acorn-globals@3.1.0
├─ acorn-jsx@3.0.1
├─ acorn@4.0.13
├─ address@1.0.3
├─ ajv-keywords@2.1.1
├─ ajv@5.5.2
├─ align-text@0.1.4
├─ alphanum-sort@1.0.2
├─ amdefine@1.0.1
├─ ansi-align@2.0.0
├─ ansi-escapes@1.4.0
├─ ansi-html@0.0.7
├─ ansi-regex@2.1.1
├─ ansi-styles@3.2.1
├─ anymatch@1.3.2
├─ append-transform@0.4.0
├─ aproba@1.2.0
├─ are-we-there-yet@1.1.4
├─ argparse@1.0.10
├─ aria-query@0.7.1

Додаток Г.


Compiled successfully!
You can now view in the browser.
http://localhost: /
http://10.112.194.139: /
Note that the development build is not optimized.
To create a production build, use yarn build.

Додаток Ґ.

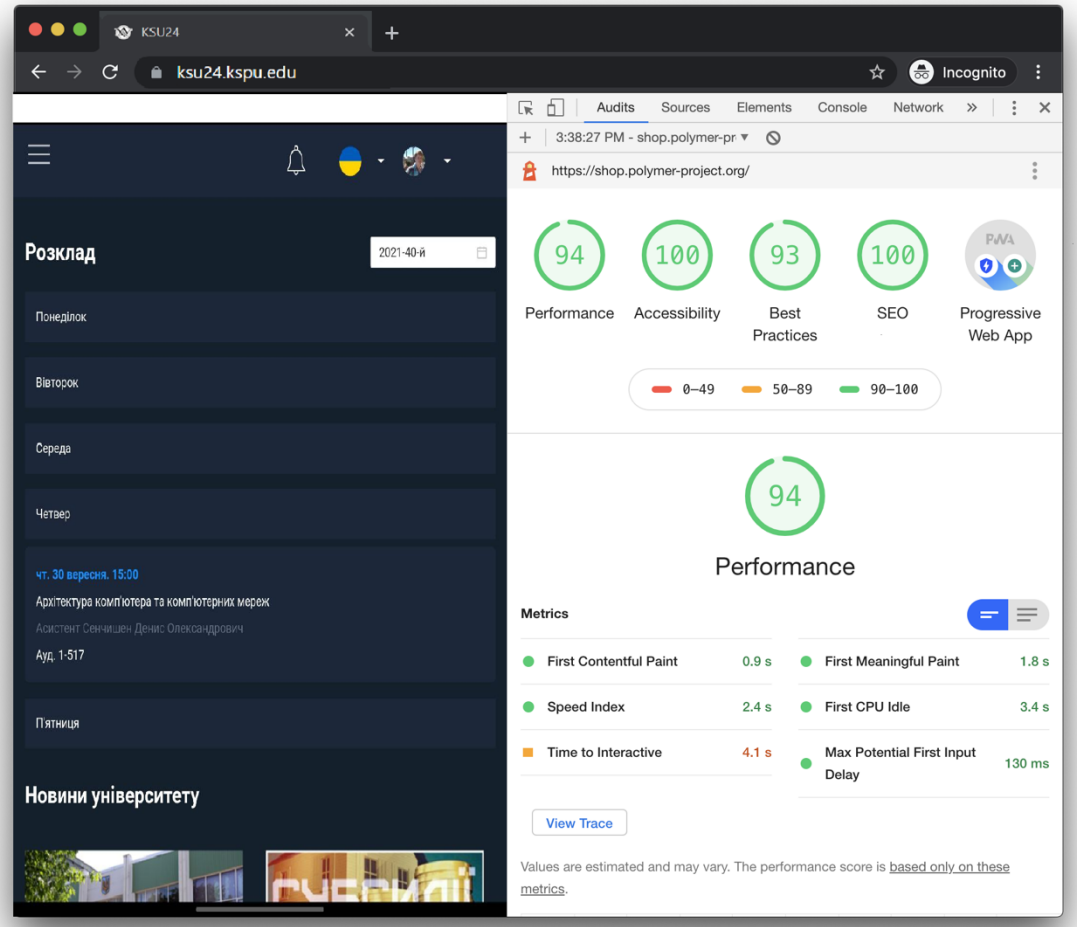

Welcome to React
To get started, edit src/App.js and save to reload.

Додаток Д.

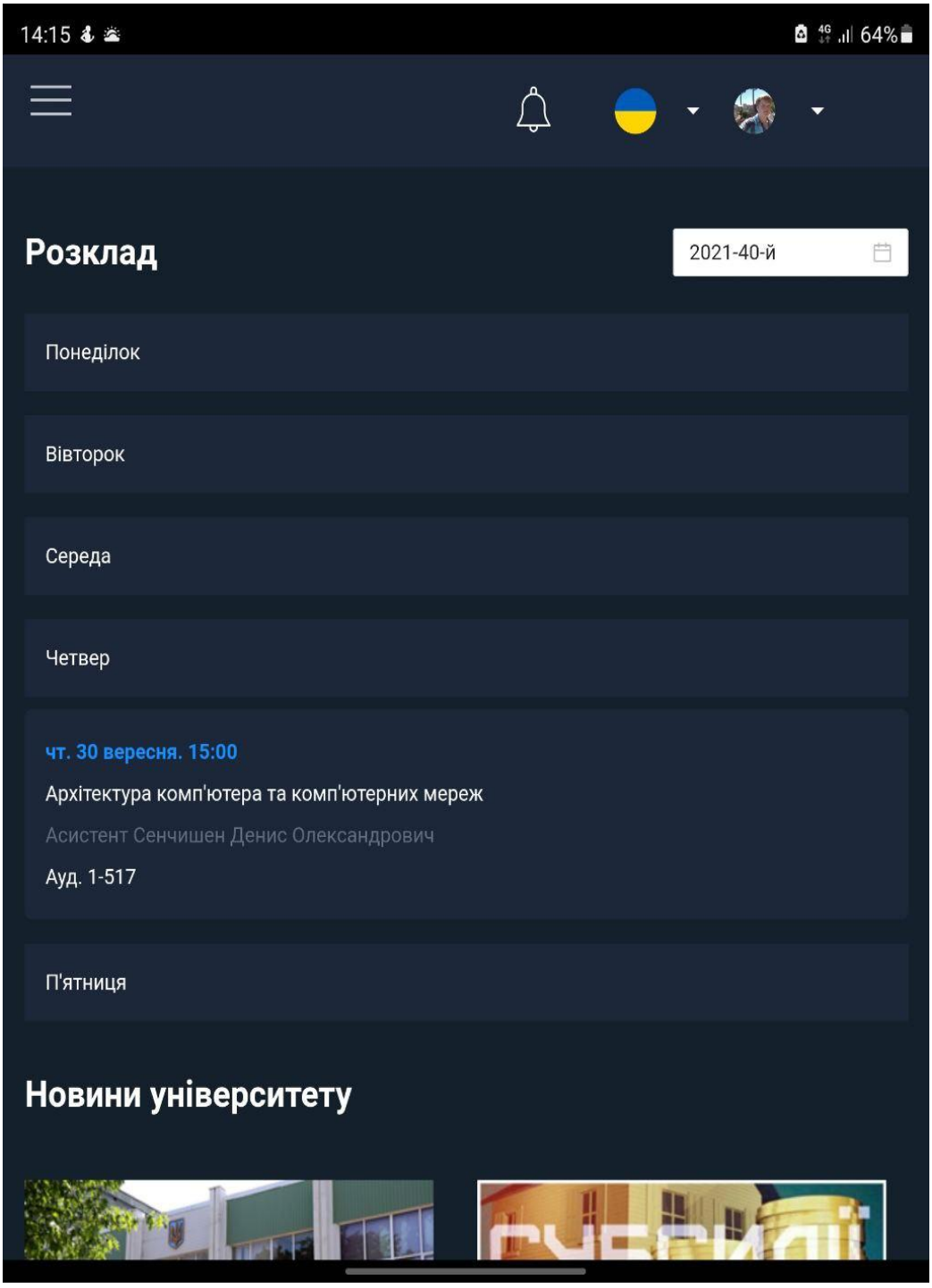
{} package.json > ...

```
{
  "name": "KSU24_front",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@babel/runtime": "^7.14.0",
    "@fullcalendar/daygrid": "^5.5.0",
    "@fullcalendar/list": "^5.5.0",
    "@fullcalendar/react": "^5.5.0",
    "@fullcalendar/resource-timegrid": "^5.5.0",
    "@react-pdf/renderer": "^2.0.19",
    "@trendmicro/react-sidenav": "^0.5.0",
    "@types/react-big-calendar": "^0.24.0",
    "@types/react-burger-menu": "^2.6.0",
    "antd": "^4.15.4",
    "axios": "^0.21.3",
    "bootstrap": "^4.4.1",
    "bootswatch": "^4.4.1",
    "i18next": "^19.9.1",
    "i18next-browser-languagedetector": "^6.0.1",
    "i18next-http-backend": "^1.1.1",
    "jwt-decode": "^3.0.0",
    "mdbreact": "^5.1.0",
    "moment": "^2.24.0",
    "node-sass": "^6.0.1",
    "plotly.js": "^1.58.4",
    "prop-types": "^15.7.2",
    "react": "^16.13.0",
    "react-big-calendar": "^0.24.1",
    "react-bootstrap": "^1.0.0-beta.16",
    "react-burger-menu": "^2.6.13",
    "react-dom": "^16.12.0",
    "react-event-calendar": "^0.3.0",
    "react-i18next": "^11.8.8",
```

Додаток Е.


KSU24
ksu24.kspu.edu
Incognito
Розклад
2021-40-й
Понеділок
Вівторок
Середа
Четвер
чт. 30 вересня. 15:00
Архітектура комп'ютера та комп'ютерних мереж
Асистент Сенчишен Денис Олександрович
Ауд. 1-517
П'ятниця
Новини університету
Audits
Sources
Elements
Console
Network
3:38:27 PM - shop.polymer-pr
https://shop.polymer-project.org/
94
Performance
100
Accessibility
93
Best Practices
100
SEO
PWA
Progressive Web App
0–49
50–89
90–100
94
Performance
Metrics
First Contentful Paint 0.9 s
First Meaningful Paint 1.8 s
Speed Index 2.4 s
First CPU Idle 3.4 s
Time to Interactive 4.1 s
Max Potential First Input Delay 130 ms
View Trace
Values are estimated and may vary. The performance score is based only on these metrics.

Додаток Є.


14:15
64%
Розклад
2021-40-й
Понеділок
Вівторок
Середа
Четвер
чт. 30 вересня. 15:00
Архітектура комп'ютера та комп'ютерних мереж
Асистент Сенчишен Денис Олександрович
Ауд. 1-517
П'ятниця
Новини університету