ДОДАТОК А

РЕЗУЛЬТАТ ПРОХОДЖЕННЯ СИСТЕМИ ПЕРЕВІРКИ ДОБРОЧЕСНОСТІ

StrikePlagiarism.com

Дата звіту 6/12/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics
Заголовок
2025_М_ПІ_ІПЗм-23-2_Дудник_О_О_скорочений
Автор
Дудник Олександр Олегович
Науковий керівник / Експерт
Каук В.І./Нечволод В.Ю.
підрозділ
каф. ПІ

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

1.26% 1.26% КП 1

3.64% 3.64% КЦ

25
Довжина фрази для коефіцієнта подібності 2

14548
Кількість слів

114660
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	1
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	3

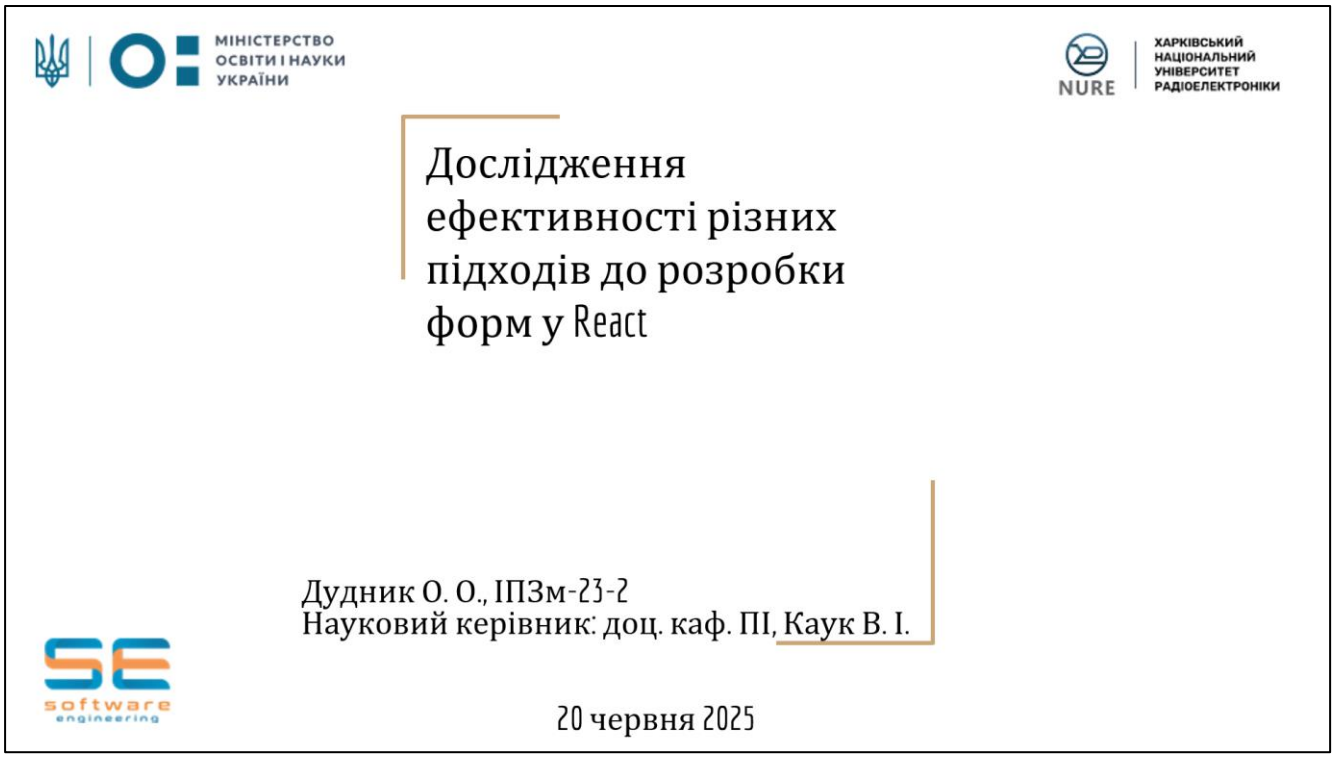
Подібності за списком джерел

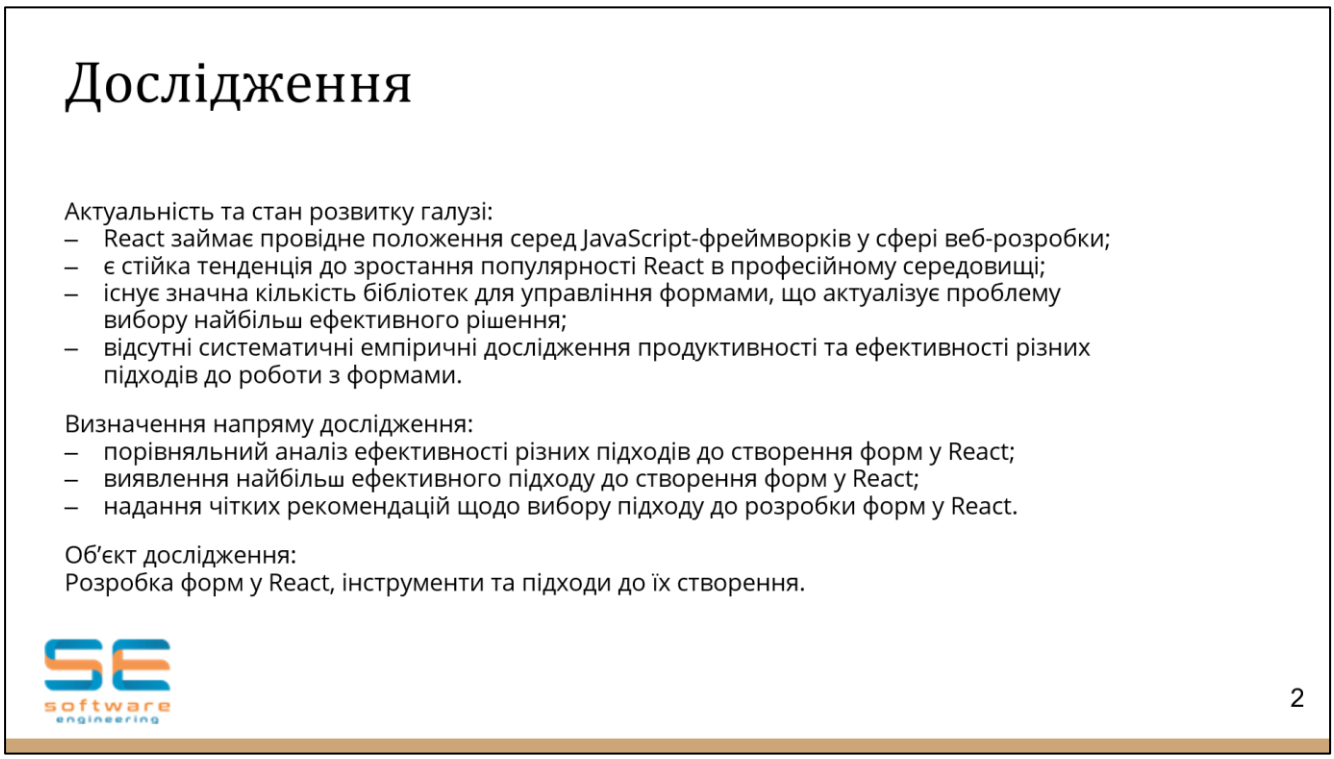
Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://community.dynamics.com/forums/thread/details/?threadid=da55fa49-c961-4b21-a761-fb43297d7bac	20 0.14 %
2	Тези доповідей ІТОНВ-2025 [redacted] 5/22/2025 Lutsk National Technical University course papers (Lutsk National Technical University course papers)	13 0.09 %

ДОДАТОК Б
СЛАЙДИ ПРЕЗЕНТАЦІЇ





Огляд літератури (аналогів)

Перелік основних джерел:

- Kainu I. Optimization in React.js: methods, tools, and techniques to improve performance of modern web applications : Бакалаврська робота. 2022. URL: https://trepo.tuni.fi/handle/10024/140258.
- de la Mora F. L., Nadi S. Which library should I use? ICSE '18: 40th International Conference on Software Engineering, м. Gothenburg Sweden. New York, NY, USA, 2018. URL: https://doi.org/10.1145/3183399.3183418.
- Kaur G., Tiwari R. G. Comparison and Analysis of Popular Frontend Frameworks and Libraries: An Evaluation of Parameters for Frontend Web Development. 2023 4th International Conference on Electronics and Sustainable Communication Systems (ICESC), м. Coimbatore, India, 6–8 лип. 2023 р. 2023. URL: https://doi.org/10.1109/icesc57686.2023.10192987.
- Sekhar Emmanni P. Comparative Analysis of Angular, React, and Vue.js in Single Page Application Development. International Journal of Science and Research (IJSR). 2023. Т. 12, № 6. С. 2971–2974. URL: https://doi.org/10.21275/sr24401230015.
- Pronina D., Kyrychenko I. Comparison of Redux and React Hooks methods in terms of performance. International Conference on Computational Linguistics and Intelligent Systems. 2022. URL: https://api.semanticscholar.org/CorpusID:250625540.

Огляд літератури (аналогів)

Актуальність джерел для дослідження:
Проаналізовані джерела охоплюють широкий спектр аспектів розробки сучасних веб-застосунків та методології їх оцінювання. Їх актуальність підтверджується такими ключовими факторами, як часова релевантність, технологічна відповідність та методологічна цінність.

Прогалини у наявних дослідженнях:
Відсутні комплексні дослідження, присвячені безпосередньо порівнянню різних підходів та бібліотек для розробки форм у React. Наявні роботи зосереджені на загальному порівнянні фреймворків або окремих аспектах розробки.

Постановка задачі

Формулювання проблеми:
Головна мета цього дослідження полягає у виявленні найбільш ефективного підходу до розробки форм у React.

Очікувані результати:
Результатом дослідження стане визначення найбільш ефективного підходу до розробки форм у React на основі багатокритеріального аналізу. Комплексна оцінка дозволить відобразити переваги та недоліки кожного з розглянутих підходів.

Методологія

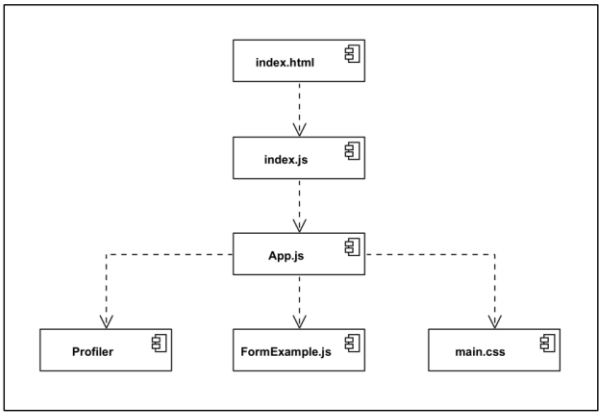
Використані методи дослідження, інструментарій та технології:

- теоретичний аналіз: аналіз предметної галузі, аналіз наукових публікацій та технічних статей, вивчення технічної документації React та досліджуваних бібліотек для розробки форм;
- експериментальне дослідження: створення тестових форм з використанням різних підходів, збір метрик продуктивності (таких як час рендерингу, обсяг пам'яті, що використовується тощо);
- багатокритеріальний аналіз: формування критеріїв для оцінки підходів, порівняння підходів за визначеними критеріями шляхом вирішення багатокритеріальної задачі прийняття рішень.

Архітектура системи для проведення експериментального дослідження

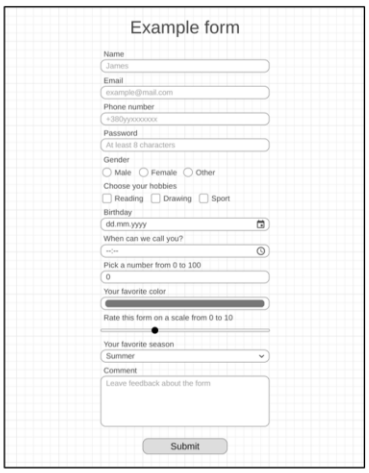
П’ять окремих тестових програм з однаковою архітектурою, кожна з яких реалізує ідентичний функціонал, але з використанням різних підходів: «чистого» React, бібліотеки Formik, бібліотеки React Hook Form, бібліотеки Final Form та бібліотеки Redux Form.

Діаграма компонентів

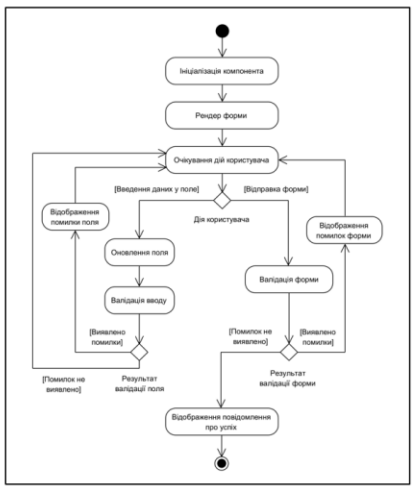


Архітектура системи для проведення експериментального дослідження

Wireframe-макет



Діаграма активностей



Опис програмного забезпечення, що було використано у дослідженні

Опис процесу розробки:

– проєктування архітектури програмного забезпечення;
– налаштування середовища розробки та конфігурація проєктів;
– програмна реалізація тестових застосунків з різними підходами.

Вибрані мови програмування та фреймворки:

– фреймворк - React;
– мова програмування - JavaScript (JSX);
– бібліотеки: Formik, React Hook Form, Final Form, Redux Form, Yup.

Зміст проведеного експерименту

Методи та послідовність:
– розробка тестових застосунків;
– вимірювання ключових показників ефективності;
– застосування методу лінійної адитивної згортки для багатокритеріального аналізу;
– визначення оптимального підходу на основі інтегрального показника ефективності.

Вхідні дані:
– досліджувані технології: React, Formik, React Hook Form, Final Form, Redux Form;
– функціональні вимоги до форми;
– критерії порівняння ефективності.

Зміст проведеного експерименту

Критерії та методи їх вимірювання:

- кількість рядків коду – підрахунок у редакторі після форматування Prettier;
- час першого рендерингу (мс) – React Profiler API;
- кількість ререндерів – React Profiler API + useEffect;
- мінімальний об’єм споживання пам’яті (Мб) – Chrome DevTools Memory Profiler;
- максимальний об’єм споживання пам’яті (Мб) – Chrome DevTools Memory Profiler;
- час відгуку INP (interaction to next paint) (мс) – Chrome DevTools, панель Performance;
- розмір бібліотеки та залежностей у бандлі (Кб) – BundlePhobia.com.

Результати експерименту

Метрики	«Чистий» React	Formik	React Hook Form	Final Form	Redux Form
Кількість рядків коду	401	358	351	418	312
Час першого рендерингу (мс)	7,6	12,4	8,9	14,8	16,5
Кількість ререндерів	137	173	4	12	183
Мінімальний об’єм споживання пам’яті (Мб)	3,7	4,2	4,1	4,2	5,7
Максимальний об’єм споживання пам’яті (Мб)	11,3	12,3	9,4	21,9	13,3
Час відгуку INP (мс)	32	40	32	40	32
Розмір бібліотеки (та залежностей) у бандлі (Кб)	0	44,8	31,3	26	99,1

Аналіз отриманих результатів

Інтерпретація результатів: значення нормалізовані та приведені за ваговими коефіцієнтами

Метрики	«Чистий» React	Formik	React Hook Form	Final Form	Redux Form	Вага
Кількість рядків коду	0,02	0,06	0,06	0,00	0,10	0,1
Час першого рендерингу	0,20	0,09	0,17	0,04	0,00	0,2
Кількість ререндерів	0,05	0,01	0,20	0,19	0,00	0,2
Мінімальний об'єм споживання пам'яті	0,10	0,08	0,08	0,08	0,00	0,1
Максимальний об'єм споживання пам'яті	0,13	0,12	0,15	0,00	0,10	0,15
Час відгуку INP	0,15	0,00	0,15	0,00	0,15	0,15
Розмір бібліотеки у бандлі	0,10	0,05	0,07	0,07	0,00	0,10
Оцінка	0,74	0,40	0,88	0,38	0,35	

Аналіз отриманих результатів

Висновки з отриманих даних:

React Hook Form демонструє найкращу загальну ефективність з оцінкою 0,88, що робить цю бібліотеку оптимальним вибором для більшості проєктів.

«Чистий» React посідає друге місце з оцінкою 0,74, що підтверджує його життєздатність для простих форм.

Formik займає третє місце з оцінкою 0,40, демонструючи середні показники серед досліджуваних бібліотек.

Final Form посідає четверте місце з оцінкою 0,38, маючи суттєві проблеми з ефективністю.

Redux Form демонструє найнижчі показники з оцінкою 0,35, що робить його найменш ефективним рішенням серед досліджуваних.

Аналіз отриманих результатів

Співставлення з цілями дослідження:

- успішно проведено комплексне порівняння п'яти підходів до розробки форм у React;
- отримано об'єктивні дані про ефективність кожного підходу за встановленими критеріями;
- визначено переваги та недоліки кожного підходу;
- надано рекомендації щодо вибору підходу залежно від специфіки проекту.

Аналіз отриманих результатів

Вплив результатів на існуючі теорії та практики:

- доповнено існуючі дослідження з порівняння бібліотек користувацького інтерфейсу в React-екосистемі специфічним аналізом засобів роботи з формами;
- створено практичну основу для прийняття обґрунтованих рішень розробниками при виборі засобів роботи з формами в React-проектах;
- продемонстровано застосовність методів багатокритеріального аналізу для порівняння програмних інструментів у веб-розробці.

Публікація результатів


MIT@AIS-2025
19 - 22 травня
Харків-Яремче 2025

Сертифікат

виданий

Олександр Дудник

за участь у
1й Міжнародній науково-технічній конференції
«Сучасні інформаційні технології та системи штучного інтелекту»
MIT@AIS-2025

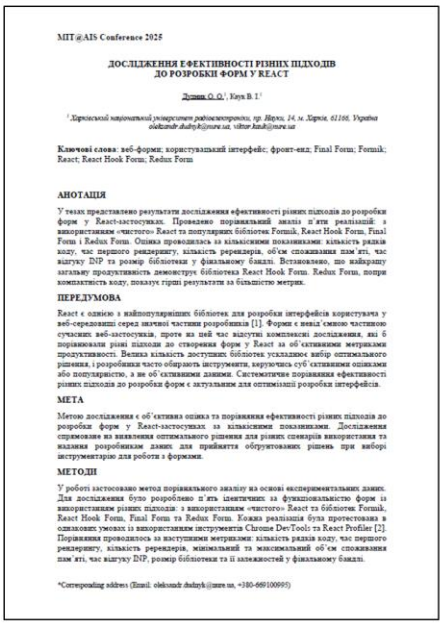
Голова конференції *Юрій РОМАНЕНКОВ*

Харків 2025

1 Міжнародна науково-практична конференція «СУЧАСНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ MIT@AIS-2025»

17

Публікація результатів


MIT@AIS Conference 2025

ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ РІЗНИХ ПІДХОДІВ ДО РОЗРОБКИ ФОРМ У REACT

Дудник О. О.[1], Клук В. І.[1]

[1] Харківський національний університет радіоелектроніки, пр. Науки, 14, м. Харків, 61166, Україна
oleksandr.dudnyk@nure.ua, viktor.kluk@nure.ua

Ключові слова: веб-форми; користувацький інтерфейс; фронт-енд; Final Form; Formik; React; React Hook Form; Redux Form

АНОТАЦІЯ

У тезах представлено результати дослідження ефективності різних підходів до розробки форм у React-застосунках. Проведено порівняльний аналіз п'яти реалізацій: з використанням «чистого» React та популярних бібліотек Formik, React Hook Form, Final Form і Redux Form. Оцінка проводилась за кількісними показниками: кількість рядків коду, час першого рендерингу, кількість рендерів, об'єм споживання пам'яті, час відгуку INP та розмір бібліотеки у фінальному бандлі. Встановлено, що найкращу загальну продуктивність демонструє бібліотека React Hook Form. Redux Form, попри компактність коду, показує гірші результати за більшістю метрик.

ПЕРЕДУМОВА

React є однією з найпопулярніших бібліотек для розробки інтерфейсів користувача у веб-середовищі серед значної частини розробників [1]. Форми є невід'ємною частиною сучасних веб-застосунків, проте на цей час відсутні комплексні дослідження, які б порівнювали різні підходи до створення форм у React за об'єктивними метриками продуктивності. Велика кількість доступних бібліотек ускладнює вибір оптимального рішення, і розробники часто обирають інструменти, керуючись суб'єктивними оцінками або популярністю, а не об'єктивними даними. Систематичне порівняння ефективності різних підходів до розробки форм є актуальним для оптимізації розробки інтерфейсів.

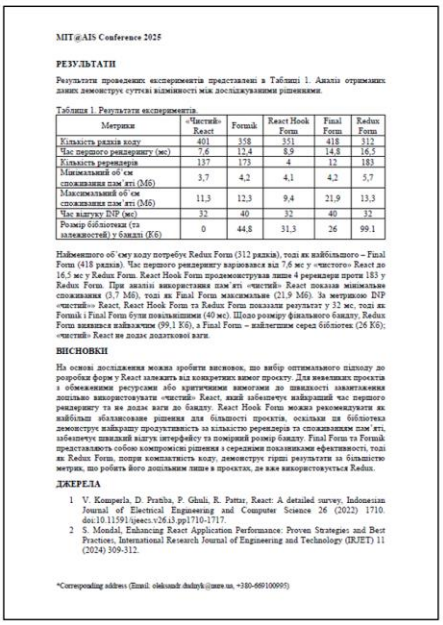
МЕТА

Метою дослідження є об'єктивна оцінка та порівняння ефективності різних підходів до розробки форм у React-застосунках за кількісними показниками. Дослідження спрямоване на виявлення оптимального рішення для різних сценаріїв використання та надання розробникам даних для прийняття обґрунтованих рішень при виборі інструментарію для роботи з формами.

МЕТОДИ

У роботі застосовано метод порівняльного аналізу на основі експериментальних даних. Для дослідження було розроблено п'ять ідентичних за функціональністю форм із використанням різних підходів: з використанням «чистого» React та бібліотек Formik, React Hook Form, Final Form та Redux Form. Кожна реалізація була протестована в однакових умовах із використанням інструменту Chrome DevTools та React Profiler [2]. Порівняння проводилось за наступними метриками: кількість рядків коду, час першого рендерингу, кількість рендерів, мінімальний та максимальний об'єм споживання пам'яті, час відгуку INP, розмір бібліотеки та її залежностей у фінальному бандлі.

*Corresponding address (Email: oleksandr.dudnyk@nure.ua, +380-669100995)


MIT@AIS Conference 2025

РЕЗУЛЬТАТИ

Результати проведених експериментів представлені в Таблиці 1. Аналіз отриманих даних демонструє суттєві відмінності між досліджуваними рішеннями.

Таблиця 1. Результати експериментів

Метрики	«Чистий» React	Formik	React Hook Form	Final Form	Redux Form
Кількість рядків коду	401	358	351	418	312
Час першого рендерингу (мс)	7,6	12,4	8,9	14,8	16,5
Кількість рендерів	137	173	4	12	183
Мінімальний об'єм споживання пам'яті (Мб)	3,7	4,2	4,1	4,2	5,7
Максимальний об'єм споживання пам'яті (Мб)	11,3	12,3	9,4	21,9	13,3
Час відгуку INP (мс)	32	40	32	40	32
Розмір бібліотеки (та залежностей) у бандлі (Кб)	0	44,8	31,3	26	99,1

Найменшого об'єму коду потребує Redux Form (312 рядків), тоді як найбільшого – Final Form (418 рядків). Час першого рендерингу варіювався від 7,6 мс у «чистого» React до 16,5 мс у Redux Form. React Hook Form продемонстрував лише 4 рендери проти 183 у Redux Form. При аналізі використання пам'яті «чистий» React показав мінімальне споживання (3,7 Мб), тоді як Final Form максимальне (21,9 Мб). За метрикою INP «чистий» React, React Hook Form та Redux Form показали результат у 32 мс, тоді як Formik і Final Form були показниками (40 мс). Щодо розміру фінального бандлу, Redux Form виявився найважчим (99,1 Кб), а Final Form – найлегшим серед бібліотек (26 Кб); «чистий» React не додає додаткової ваги.

ВИСНОВКИ

На основі дослідження можна зробити висновок, що вибір оптимального підходу до розробки форм у React залежить від конкретних вимог проєкту. Для невеликих проєктів з обмеженими ресурсами або критичними вимогами до швидкості завантаження доцільно використовувати «чистий» React, який забезпечує найкращий час першого рендерингу та не додає ваги до бандлу. React Hook Form можна рекомендувати як найбільш збалансоване рішення для більшості проєктів, оскільки ця бібліотека демонструє найкращу продуктивність за кількістю рендерів та споживанням пам'яті, забезпечує швидкий відгук інтерфейсу та помірний розмір бандлу. Final Form та Formik представляють собою компромісні рішення з середніми показниками ефективності, тоді як Redux Form, попри компактність коду, демонструє гірші результати за більшістю метрик, що робить його доцільним лише в проєктах, де вже використовується Redux.

ДЖЕРЕЛА

1. V. Komperla, D. Prathiba, P. Ghuli, R. Pattar, React: A detailed survey, Indonesian Journal of Electrical Engineering and Computer Science 26 (2022) 1710. doi:10.11591/ijeecs.v26.i3.pp1710-1717.
2. S. Mondal, Enhancing React Application Performance: Proven Strategies and Best Practices, International Research Journal of Engineering and Technology (IRJET) 11 (2024) 309-312.

*Corresponding address (Email: oleksandr.dudnyk@nure.ua, +380-669100995)

18

Підсумки

Отримані результати можуть бути використані розробниками та архітекторами програмного забезпечення при виборі оптимального підходу до реалізації форм у React-застосунках.

Подальші дослідження можуть бути спрямовані на розширення набору критеріїв оцінки, включення інших популярних бібліотек для роботи з формами, а також на дослідження впливу різних факторів на ефективність різних підходів.

ДОДАТОК В

АПРОБАЦІЯ РЕЗУЛЬТАТІВ РОБОТИ

1 Міжнародна науково-практична конференція «СУЧАСНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ MIT@AIS-2025»

ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ РІЗНИХ ПІДХОДІВ ДО РОЗРОБКИ ФОРМ У REACT

Дудник О. О.[1], Каук В. І.[1]

[1] *Харківський національний університет радіоелектроніки, пр. Науки, 14, м. Харків, 61166, Україна*
oleksandr.dudnyk@nure.ua, viktor.kauk@nure.ua

Ключові слова: веб-форми; користувацький інтерфейс; фронт-енд; Final Form; Formik; React; React Hook Form; Redux Form

АНОТАЦІЯ

У тезах представлено результати дослідження ефективності різних підходів до розробки форм у React-застосунках. Проведено порівняльний аналіз п’яти реалізацій: з використанням «чистого» React та популярних бібліотек Formik, React Hook Form, Final Form і Redux Form. Оцінка проводилась за кількісними показниками: кількість рядків коду, час першого рендерингу, кількість ререндерів, об'єм споживання пам’яті, час відгуку INP та розмір бібліотеки у фінальному бандлі. Встановлено, що найкращу загальну продуктивність демонструє бібліотека React Hook Form. Redux Form, попри компактність коду, показує гірші результати за більшістю метрик.

ПЕРЕДУМОВА

React є однією з найпопулярніших бібліотек для розробки інтерфейсів користувача у веб-середовищі серед значної частини розробників [1]. Форми є невід’ємною частиною сучасних веб-застосунків, проте на цей час відсутні комплексні дослідження, які б порівнювали різні підходи до створення форм у React за об’єктивними метриками продуктивності. Велика кількість доступних бібліотек ускладнює вибір оптимального рішення, і розробники часто обирають інструменти, керуючись суб’єктивними оцінками або популярністю, а не об’єктивними даними. Систематичне порівняння ефективності різних підходів до розробки форм є актуальним для оптимізації розробки інтерфейсів.

МЕТА

Метою дослідження є об’єктивна оцінка та порівняння ефективності різних підходів до розробки форм у React-застосунках за кількісними показниками. Дослідження спрямоване на виявлення оптимального рішення для різних сценаріїв використання та надання розробникам даних для прийняття обґрунтованих рішень при виборі інструментарію для роботи з формами.

МЕТОДИ

У роботі застосовано метод порівняльного аналізу на основі експериментальних даних. Для дослідження було розроблено п’ять ідентичних за функціональністю форм із використанням різних підходів: з використанням «чистого» React та бібліотек Formik, React Hook Form, Final Form та Redux Form. Кожна реалізація була протестована в однакових умовах із використанням інструментів Chrome DevTools та React Profiler [2]. Порівняння проводилось за наступними метриками: кількість рядків коду, час першого рендерингу, кількість ререндерів, мінімальний та максимальний об’єм споживання пам’яті, час відгуку INP, розмір бібліотеки та її залежностей у фінальному бандлі.

РЕЗУЛЬТАТИ

Результати проведених експериментів представлені в Таблиці 1. Аналіз отриманих даних демонструє суттєві відмінності між досліджуваними рішеннями.

Таблиця 1. Результати експериментів.

Метрики	«Чистий» React	Formik	React Hook Form	Final Form	Redux Form
Кількість рядків коду	401	358	351	418	312
Час першого рендерингу (мс)	7,6	12,4	8,9	14,8	16,5
Кількість ререндерів	137	173	4	12	183
Мінімальний об’єм споживання пам’яті (Мб)	3,7	4,2	4,1	4,2	5,7
Максимальний об’єм споживання пам’яті (Мб)	11,3	12,3	9,4	21,9	13,3
Час відгуку INP (мс)	32	40	32	40	32
Розмір бібліотеки (та залежностей) у бандлі (Кб)	0	44,8	31,3	26	99.1

Найменшого об’єму коду потребує Redux Form (312 рядків), тоді як найбільшого – Final Form (418 рядків). Час першого рендерингу варіювався від 7,6 мс у «чистого» React до 16,5 мс у Redux Form. React Hook Form продемонстрував лише 4 ререндери проти 183 у Redux Form. При аналізі використання пам’яті «чистий» React показав мінімальне споживання (3,7 Мб), тоді як Final Form максимальне (21,9 Мб). За метрикою INP «чистий»» React, React Hook Form та Redux Form показали результат у 32 мс, тоді як Formik і Final Form були повільнішими (40 мс). Щодо розміру фінального бандлу, Redux Form виявився найважчим (99,1 Кб), а Final Form – найлегшим серед бібліотек (26 Кб); «чистий» React не додає додаткової ваги.

ВИСНОВКИ

На основі дослідження можна зробити висновок, що вибір оптимального підходу до розробки форм у React залежить від конкретних вимог проєкту. Для невеликих проєктів з обмеженими ресурсами або критичними вимогами до швидкості завантаження доцільно використовувати «чистий» React, який забезпечує найкращий час першого рендерингу та не додає ваги до бандлу. React Hook Form можна рекомендувати як найбільш збалансоване рішення для більшості проєктів, оскільки ця бібліотека демонструє найкращу продуктивність за кількістю ререндерів та споживанням пам’яті, забезпечує швидкий відгук інтерфейсу та помірний розмір бандлу. Final Form та Formik представляють собою компромісні рішення з середніми показниками ефективності, тоді як Redux Form, попри компактність коду, демонструє гірші результати за більшістю метрик, що робить його доцільним лише в проєктах, де вже використовується Redux.

ДЖЕРЕЛА

1 V. Komperla, D. Pratiba, P. Ghuli, R. Pattar, React: A detailed survey, Indonesian Journal of Electrical Engineering and Computer Science 26 (2022) 1710. doi:10.11591/ijeecs.v26.i3.pp1710-1717.
2 S. Mondal, Enhancing React Application Performance: Proven Strategies and Best Practices, International Research Journal of Engineering and Technology (IRJET) 11 (2024) 309-312.

ДОДАТОК Г

ЕКСПЕРТНИЙ ВИСНОВОК РЕЗУЛЬТАТІВ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА ВІДПОВІДНІСТЬ ОФОРМЛЕННЯ ВИМОГАМ ДСТУ 3008:2015

Експертний висновок результатів перевірки кваліфікаційної роботи

студент (посада) | програмної інженерії (кафедра) | ІПЗм-23-2 (група)

Олександр ДУДНИК
(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
	7.3 Нумерація сторінок звіту	
	7.5 Рисунки	
	7.6 Таблиці	
	7.7 Переліки	
	7.8 Примітки	
	7.9 Виноски	
	7.10 Формули та рівняння	
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні познаки	
	7.15 Додатки	

Експерт _____ (підпис) Вадим НЕЧВОЛОД (прізвище, ініціали)

Зауважень немає.Вкажіть джерела походження рисунків.
12.06.2025

ДОДАТОК Д

КОД ФАЙЛУ MAIN.CSS

```
1. /*================ GLOBAL ================*/
2.
3. @import
   url('https://fonts.googleapis.com/css2?family=Montserrat:ital,wght@
   0,100..900;1,100..900&display=swap');
4.
5. /*font-family: 'Montserrat', sans-serif;*/
6.
7. :root {
8.     --color-primary: #EDBB36;
9.
10.       --color-background: #FCF9E4;
11.
12.       --color-button: #F7CF6B;
13.       --color-button-highlight: #F6C447;
14.
15.       --color-text: #4E2B11;
16.       --color-text-placeholder: #C3A997;
17.
18.       --color-error: #E11C25;
19.   }
20.
21.   /*================ RESET STYLES ================*/
22.
23.   * {
24.       box-sizing: border-box;
25.   }
26.
27.   :active, :hover, :focus {
28.       outline: 0;
29.       outline-offset: 0;
30.   }
31.
32.   button {
33.       border: none;
34.       background: none;
35.       margin: 0;
36.       padding: 0;
37.   }
38.
39.   h1, h2, h3, h4, h5, h6 {
40.       font-weight: normal;
41.   }
42.
43.   /*================ LAYOUT ================*/
44.
45.   .form-window {
46.       background-color: var(--color-background);
47.       height: 100vh;
48.       width: 100vw;
49.
50.       overflow: auto;
51.       scrollbar-gutter: stable both-edges;
```

```
52.   }
53.
54.   .form-container {
55.       width: 30vw;
56.       margin: auto;
57.
58.       padding: 5vw 0;
59.   }
60.
61.   .form-title {
62.       font-family: 'Montserrat', sans-serif;
63.       font-size: 3.6vw;
64.       color: var(--color-text);
65.       text-align: center;
66.
67.       margin-bottom: 2vw;
68.   }
69.
70.   /*================ FIELDS ================*/
71.
72.   .form-label {
73.       display: block;
74.       margin: 0.8vw 0.3vw 0.2vw;
75.
76.       font-family: 'Montserrat', sans-serif;
77.       font-size: 1.4vw;
78.       font-weight: 500;
79.       color: var(--color-text);
80.   }
81.
82.   .form-field, .form-textarea {
83.       display: block;
84.       width: 100%;
85.       padding: 0.2vw 1vw;
86.
87.       border: var(--color-primary) solid 0.2vw;
88.       border-radius: 1vw;
89.       background-color: var(--color-background);
90.
91.       font-family: 'Montserrat', sans-serif;
92.       font-size: 1.4vw;
93.       font-weight: 500;
94.       color: var(--color-text);
95.   }
96.
97.   .form-field::placeholder, .form-textarea::placeholder {
98.       color: var(--color-text-placeholder);
99.   }
100.
101.  .form-textarea {
102.      height: 10vw;
103.      resize: none;
104.  }
105.
106.  /*================ BUTTONS ================*/
107.
108.  .form-button {
109.      display: block;
```

```
110.
111.     background-color: var(--color-button);
112.     border-radius: 1vw;
113.
114.     padding: 0.3vw 5vw;
115.     margin: 2vw auto 0;
116.
117.     font-family: 'Montserrat', sans-serif;
118.     font-size: 1.6vw;
119.     color: var(--color-text);
120.
121.     cursor: pointer;
122. }
123.
124. .form-button:hover {
125.     background-color: var(--color-button-highlight);
126. }
127.
128. /*================ SELECTORS ================*/
129.
130. .form-selection-wrapper {
131.     display: inline-block;
132.     margin: 0.3vw;
133. }
134.
135. .form-selection-label {
136.     font-family: 'Montserrat', sans-serif;
137.     font-size: 1.4vw;
138.     font-weight: 500;
139.     color: var(--color-text);
140.
141.     margin: 0 0.5vw;
142.
143.     vertical-align: middle;
144. }
145.
146. .form-checkbox, .form-radio {
147.     width: 1.4vw;
148.     height: 1.4vw;
149.     margin: 0;
150.
151.     vertical-align: middle;
152. }
153.
154. /*================ ERROR MESSAGE ================*/
155.
156. .form-error-message {
157.     display: block;
158.     margin: 0.6vw 0.3vw 1.2vw;
159.
160.     font-family: 'Montserrat', sans-serif;
161.     font-size: 1.1vw;
162.     font-weight: 500;
163.     color: var(--color-error);
164. }
165.
166. .form-error-message:empty {
167.     display: none;
```

```
168.  }
169.
170.  /*================= BROWSER TWEAKS ================*/
171.
172.  * {
173.      accent-color: var(--color-primary);
174.  }
175.
176.  ::-webkit-calendar-picker-indicator {
177.      filter: invert(78%) sepia(38%) saturate(1011%) hue-
   rotate(356deg) brightness(98%) contrast(92%);
178.  }
```

ДОДАТОК Е

КОД ФАЙЛУ FORMEXAMPLE.JS («ЧИСТИЙ» REACT)

```
1. import React, { useState, useRef } from "react";
2. import * as yup from "yup";
3.
4. function FormExample() {
5.     const [formData, setFormData] = useState({
6.         name: "",
7.         email: "",
8.         phone: "",
9.         password: "",
10.           gender: "",
11.           hobbyReading: false,
12.           hobbyDrawing: false,
13.           hobbySport: false,
14.           birthday: "",
15.           time: "",
16.           number: 0,
17.           color: "#EDBB36",
18.           rating: 10,
19.           season: "summer",
20.           comment: "",
21.       });
22.
23.       const [fieldErrors, setFieldErrors] = useState({});
24.
25.       const validateField = async (name, value) => {
26.           try {
27.               await yup.reach(validationSchema,
  name).validate(value);
28.               setFieldErrors((prev) => ({
29.                   ...prev,
30.                   [name]: undefined,
31.               }));
32.           } catch (err) {
33.               setFieldErrors((prev) => ({
34.                   ...prev,
35.                   [name]: err.message,
36.               }));
37.           }
38.       };
39.
40.       const handleChange = (event) => {
41.           const { name, value } = event.target;
42.           setFormData((prev) => ({
43.               ...prev,
44.               [name]: value,
45.           }));
46.           validateField(name, value);
47.       };
48.
49.       const handleCheckboxChange = (event) => {
50.           const { name, checked } = event.target;
51.           setFormData((prev) => ({
52.               ...prev,
```

```
53.                 [name]: checked,
54.             }));
55.         };
56.
57.         const validationSchema = yup.object().shape({
58.             name: yup
59.                 .string()
60.                 .matches(/^[A-Za-z]+$/, "Name must contain only
   English letters")
61.                 .required("Name is required"),
62.             email: yup
63.                 .string()
64.                 .email("Invalid email format")
65.                 .required("Email is required"),
66.             phone: yup
67.                 .string()
68.                 .matches(
69.                     /^\+380\d{9}$/,
70.                     "Phone number must start with '+380' and contain
   12 digits"
71.                 )
72.                 .required("Phone number is required"),
73.             password: yup
74.                 .string()
75.                 .min(8, "Password must be at least 8 characters
   long")
76.                 .matches(
77.                     /[a-z]/,
78.                     "Password must contain at least one lowercase
   letter"
79.                 )
80.                 .matches(
81.                     /[A-Z]/,
82.                     "Password must contain at least one uppercase
   letter"
83.                 )
84.                 .matches(/\d/, "Password must contain at least one
   number")
85.                 .matches(
86.                     /[@$!%*?&#]/,
87.                     "Password must contain at least one special
   character"
88.                 )
89.                 .required("Password is required"),
90.             gender: yup.string().required("Gender is required"),
91.             birthday: yup
92.                 .date()
93.                 .required("Birthday is required")
94.                 .test(
95.                     "age",
96.                     "You must be at least 13 years old and not older
   than 100 years (because it is impossible)",
97.                     (value) => {
98.                         if (!value) return false;
99.                         const today = new Date();
100.                        const birthDate = new Date(value);
101.                        const age = today.getFullYear() -
   birthDate.getFullYear();
```

```
                    const monthDiff = today.getMonth() -
  birthDate.getMonth();
                    if (
                        monthDiff < 0 ||
                        (monthDiff === 0 &&
                            today.getDate() <
  birthDate.getDate())
                    ) {
                        return age - 1 >= 13 && age - 1 <= 100;
                    }
                    return age >= 13 && age <= 100;
                }
            ),
        time: yup.string().required("Time is required"),
        number: yup
            .number()
            .required("Number is required")
            .min(0, "Number must be at least 0")
            .max(100, "Number must be at most 100"),
    });

    const handleSubmit = async () => {
        try {
            await validationSchema.validate(formData, {
  abortEarly: false });
            alert("Form submitted successfully!");
            setFieldErrors({});
        } catch (err) {
            if (err instanceof yup.ValidationError) {
                const errors = {};
                err.inner.forEach((error) => {
                    errors[error.path] = error.message;
                });
                setFieldErrors(errors);
            }
        }
    };

    return (
        <div className="form-window">
            <div className="form-container">
                <div className="form-title">Example form</div>

                {/* Name */}
                <label className="form-label" htmlFor="name">
                    Name
                </label>
                <input
                    className="form-field"
                    type="text"
                    id="name"
                    name="name"
                    value={formData.name}
                    onChange={handleChange}
                    placeholder="James"
                />
                {fieldErrors.name && (
```

```
156.                     <div className="form-error-
  message">{fieldErrors.name}</div>
157.                 )}
158.
159.                 {/* Email */}
160.                 <label className="form-label" htmlFor="email">
161.                     Email
162.                 </label>
163.                 <input
164.                     className="form-field"
165.                     type="text"
166.                     id="email"
167.                     name="email"
168.                     value={formData.email}
169.                     onChange={handleChange}
170.                     placeholder="example@mail.com"
171.                 />
172.                 {fieldErrors.email && (
173.                     <div className="form-error-message">
174.                         {fieldErrors.email}
175.                     </div>
176.                 )}
177.
178.                 {/* Phone */}
179.                 <label className="form-label" htmlFor="phone">
180.                     Phone number
181.                 </label>
182.                 <input
183.                     className="form-field"
184.                     type="text"
185.                     id="phone"
186.                     name="phone"
187.                     value={formData.phone}
188.                     onChange={handleChange}
189.                     placeholder="+380yyxxxxxxx"
190.                 />
191.                 {fieldErrors.phone && (
192.                     <div className="form-error-message">
193.                         {fieldErrors.phone}
194.                     </div>
195.                 )}
196.
197.                 {/* Password */}
198.                 <label className="form-label"
  htmlFor="password">
199.                     Password
200.                 </label>
201.                 <input
202.                     className="form-field"
203.                     type="password"
204.                     id="password"
205.                     name="password"
206.                     value={formData.password}
207.                     onChange={handleChange}
208.                     placeholder="At least 8 characters"
209.                 />
210.                 {fieldErrors.password && (
211.                     <div className="form-error-message">
```

```
                        {fieldErrors.password}
                    </div>
                )}

                {/* Gender */}
                <div className="form-label">Gender</div>

                <div className="form-selection-wrapper">
                    <input
                        className="form-radio"
                        type="radio"
                        id="male"
                        name="gender"
                        value="male"
                        checked={formData.gender === "male"}
                        onChange={handleChange}
                    />
                    <label className="form-selection-label"
  htmlFor="male">
                        Male
                    </label>
                </div>

                <div className="form-selection-wrapper">
                    <input
                        className="form-radio"
                        type="radio"
                        id="female"
                        name="gender"
                        value="female"
                        checked={formData.gender === "female"}
                        onChange={handleChange}
                    />
                    <label className="form-selection-label"
  htmlFor="female">
                        Female
                    </label>
                </div>

                <div className="form-selection-wrapper">
                    <input
                        className="form-radio"
                        type="radio"
                        id="other"
                        name="gender"
                        value="other"
                        checked={formData.gender === "other"}
                        onChange={handleChange}
                    />
                    <label className="form-selection-label"
  htmlFor="other">
                        Other
                    </label>
                </div>
                {fieldErrors.gender && (
                    <div className="form-error-message">
                        {fieldErrors.gender}
                    </div>
```

```
267.                 )}
268.
269.                 {/* Hobbies */}
270.                 <div className="form-label">Choose your
   hobbies</div>
271.
272.                 <div className="form-selection-wrapper">
273.                     <input
274.                         className="form-checkbox"
275.                         type="checkbox"
276.                         id="hobbyReading"
277.                         name="hobbyReading"
278.                         checked={formData.hobbyReading}
279.                         onChange={handleCheckboxChange}
280.                     />
281.                     <label
282.                         className="form-selection-label"
283.                         htmlFor="hobbyReading"
284.                     >
285.                         Reading
286.                     </label>
287.                 </div>
288.
289.                 <div className="form-selection-wrapper">
290.                     <input
291.                         className="form-checkbox"
292.                         type="checkbox"
293.                         id="hobbyDrawing"
294.                         name="hobbyDrawing"
295.                         checked={formData.hobbyDrawing}
296.                         onChange={handleCheckboxChange}
297.                     />
298.                     <label
299.                         className="form-selection-label"
300.                         htmlFor="hobbyDrawing"
301.                     >
302.                         Drawing
303.                     </label>
304.                 </div>
305.
306.                 <div className="form-selection-wrapper">
307.                     <input
308.                         className="form-checkbox"
309.                         type="checkbox"
310.                         id="hobbySport"
311.                         name="hobbySport"
312.                         checked={formData.hobbySport}
313.                         onChange={handleCheckboxChange}
314.                     />
315.                     <label
316.                         className="form-selection-label"
317.                         htmlFor="hobbySport"
318.                     >
319.                         Sport
320.                     </label>
321.                 </div>
322.
323.                 {/* Birthday */}
```

```
                <label className="form-label"
  htmlFor="birthday">
                    Birthday
                </label>
                <input
                    className="form-field"
                    type="date"
                    id="birthday"
                    name="birthday"
                    value={formData.birthday}
                    onChange={handleChange}
                />
                {fieldErrors.birthday && (
                    <div className="form-error-message">
                        {fieldErrors.birthday}
                    </div>
                )}

                {/* Time */}
                <label className="form-label" htmlFor="time">
                    When can we call you?
                </label>
                <input
                    className="form-field"
                    type="time"
                    id="time"
                    name="time"
                    value={formData.time}
                    onChange={handleChange}
                />
                {fieldErrors.time && (
                    <div className="form-error-
  message">{fieldErrors.time}</div>
                )}

                {/* Number */}
                <label className="form-label" htmlFor="number">
                    Pick a number from 0 to 100
                </label>
                <input
                    className="form-field"
                    type="number"
                    id="number"
                    name="number"
                    value={formData.number}
                    onChange={handleChange}
                    min="0"
                    max="100"
                />
                {fieldErrors.number && (
                    <div className="form-error-message">
                        {fieldErrors.number}
                    </div>
                )}

                {/* Color */}
                <label className="form-label" htmlFor="color">
                    Your favorite color
```

```
                </label>
                <input
                    className="form-field"
                    type="color"
                    id="color"
                    name="color"
                    value={formData.color}
                    onChange={handleChange}
                />

                {/* Rating */}
                <label className="form-label" htmlFor="rating">
                    Rate this form on a scale from 0 to 10
                </label>
                <input
                    className="form-field"
                    type="range"
                    id="rating"
                    name="rating"
                    value={formData.rating}
                    onChange={handleChange}
                    min="0"
                    max="10"
                />

                {/* Season */}
                <label className="form-label" htmlFor="season">
                    Your favorite season
                </label>
                <select
                    className="form-field"
                    id="season"
                    name="season"
                    value={formData.season}
                    onChange={handleChange}
                >
                    <option value="summer">Summer</option>
                    <option value="autumn">Autumn</option>
                    <option value="winter">Winter</option>
                    <option value="spring">Spring</option>
                </select>

                {/* Comment */}
                <label className="form-label" htmlFor="comment">
                    Comment
                </label>
                <textarea
                    className="form-textarea"
                    id="comment"
                    name="comment"
                    value={formData.comment}
                    onChange={handleChange}
                    placeholder="Leave feedback about the form"
                />

                {/* Submit */}
                <button className="form-button"
  onClick={handleSubmit}>
```

```
                    Submit
                  </button>
                </div>
              </div>
    );
}

export default FormExample;
```

ДОДАТОК Ж

КОД ФАЙЛУ FORMEXAMPLE.JS (FORMIK)

```
1. import React, { useRef } from "react";
2. import { Formik, Form, Field, ErrorMessage } from "formik";
3. import * as yup from "yup";
4.
5. function FormExample() {
6.
7.     const initialValues = {
8.         name: "",
9.         email: "",
10.          phone: "",
11.          password: "",
12.          gender: "",
13.          hobbyReading: false,
14.          hobbyDrawing: false,
15.          hobbySport: false,
16.          birthday: "",
17.          time: "",
18.          number: 0,
19.          color: "#EDBB36",
20.          rating: 10,
21.          season: "summer",
22.          comment: "",
23.       };
24.
25.       const validationSchema = yup.object().shape({
26.          name: yup
27.              .string()
28.              .matches(/^[A-Za-z]+$/, "Name must contain only
   English letters")
29.              .required("Name is required"),
30.          email: yup
31.              .string()
32.              .email("Invalid email format")
33.              .required("Email is required"),
34.          phone: yup
35.              .string()
36.              .matches(
37.                  /^\+380\d{9}$/,
38.                  "Phone number must start with '+380' and contain
   12 digits"
39.              )
40.              .required("Phone number is required"),
41.          password: yup
42.              .string()
43.              .min(8, "Password must be at least 8 characters
   long")
44.              .matches(
45.                  /[a-z]/,
46.                  "Password must contain at least one lowercase
   letter"
47.              )
48.              .matches(
49.                  /[A-Z]/,
```

```
                    "Password must contain at least one uppercase
  letter"
                )
                .matches(/\d/, "Password must contain at least one
  number")
                .matches(
                    /[@$!%*?&#]/,
                    "Password must contain at least one special
  character"
                )
                .required("Password is required"),
            gender: yup.string().required("Gender is required"),
            birthday: yup
                .date()
                .required("Birthday is required")
                .test(
                    "age",
                    "You must be at least 13 years old and not older
  than 100 years (because it is impossible)",
                    (value) => {
                        if (!value) return false;
                        const today = new Date();
                        const birthDate = new Date(value);
                        const age = today.getFullYear() -
  birthDate.getFullYear();
                        const monthDiff = today.getMonth() -
  birthDate.getMonth();
                        if (
                            monthDiff < 0 ||
                            (monthDiff === 0 &&
                                today.getDate() <
  birthDate.getDate())
                        ) {
                            return age - 1 >= 13 && age - 1 <= 100;
                        }
                        return age >= 13 && age <= 100;
                    }
                ),
            time: yup.string().required("Time is required"),
            number: yup
                .number()
                .required("Number is required")
                .min(0, "Number must be at least 0")
                .max(100, "Number must be at most 100"),
        });

        const handleSubmit = (values, { setSubmitting }) => {
            alert("Form submitted successfully!");
            setSubmitting(false);
        };

        return (
            <div className="form-window">
                <div className="form-container">
                    <div className="form-title">Example form</div>
                    <Formik
                        initialValues={initialValues}
                        validationSchema={validationSchema}
```

```
                    onSubmit={handleSubmit}
                >
                    {({ isSubmitting }) => (
                        <Form>
                            {/* Name */}
                            <label className="form-label"
  htmlFor="name">
                                Name
                            </label>
                            <Field
                                className="form-field"
                                type="text"
                                id="name"
                                name="name"
                                placeholder="James"
                            />
                            <ErrorMessage
                                className="form-error-message"
                                name="name"
                                component="div"
                            />

                            {/* Email */}
                            <label className="form-label"
  htmlFor="email">
                                Email
                            </label>
                            <Field
                                className="form-field"
                                type="text"
                                id="email"
                                name="email"
                                placeholder="example@mail.com"
                            />
                            <ErrorMessage
                                className="form-error-message"
                                name="email"
                                component="div"
                            />

                            {/* Phone */}
                            <label className="form-label"
  htmlFor="phone">
                                Phone number
                            </label>
                            <Field
                                className="form-field"
                                type="text"
                                id="phone"
                                name="phone"
                                placeholder="+380yyxxxxxxx"
                            />
                            <ErrorMessage
                                className="form-error-message"
                                name="phone"
                                component="div"
                            />

```

```
156.                                {/* Password */}
157.                                <label className="form-label"
   htmlFor="password">
158.                                    Password
159.                                </label>
160.                                <Field
161.                                    className="form-field"
162.                                    type="password"
163.                                    id="password"
164.                                    name="password"
165.                                    placeholder="At least 8
   characters"
166.                                />
167.                                <ErrorMessage
168.                                    className="form-error-message"
169.                                    name="password"
170.                                    component="div"
171.                                />
172.
173.                                {/* Gender */}
174.                                <div className="form-
   label">Gender</div>
175.                                <div className="form-selection-
   wrapper">
176.                                    <Field
177.                                        className="form-radio"
178.                                        type="radio"
179.                                        id="male"
180.                                        name="gender"
181.                                        value="male"
182.                                    />
183.                                    <label
184.                                        className="form-selection-
   label"
185.                                        htmlFor="male"
186.                                    >
187.                                        Male
188.                                    </label>
189.                                </div>
190.                                <div className="form-selection-
   wrapper">
191.                                    <Field
192.                                        className="form-radio"
193.                                        type="radio"
194.                                        id="female"
195.                                        name="gender"
196.                                        value="female"
197.                                    />
198.                                    <label
199.                                        className="form-selection-
   label"
200.                                        htmlFor="female"
201.                                    >
202.                                        Female
203.                                    </label>
204.                                </div>
205.                                <div className="form-selection-
   wrapper">
```

```
206.                                    <Field
207.                                        className="form-radio"
208.                                        type="radio"
209.                                        id="other"
210.                                        name="gender"
211.                                        value="other"
212.                                    />
213.                                    <label
214.                                        className="form-selection-
   label"
215.                                        htmlFor="other"
216.                                    >
217.                                        Other
218.                                    </label>
219.                                </div>
220.                                <ErrorMessage
221.                                    className="form-error-message"
222.                                    name="gender"
223.                                    component="div"
224.                                />
225.
226.                                {/* Hobbies */}
227.                                <div className="form-label">
228.                                    Choose your hobbies
229.                                </div>
230.                                <div className="form-selection-
   wrapper">
231.                                    <Field
232.                                        className="form-checkbox"
233.                                        type="checkbox"
234.                                        id="hobbyReading"
235.                                        name="hobbyReading"
236.                                    />
237.                                    <label
238.                                        className="form-selection-
   label"
239.                                        htmlFor="hobbyReading"
240.                                    >
241.                                        Reading
242.                                    </label>
243.                                </div>
244.                                <div className="form-selection-
   wrapper">
245.                                    <Field
246.                                        className="form-checkbox"
247.                                        type="checkbox"
248.                                        id="hobbyDrawing"
249.                                        name="hobbyDrawing"
250.                                    />
251.                                    <label
252.                                        className="form-selection-
   label"
253.                                        htmlFor="hobbyDrawing"
254.                                    >
255.                                        Drawing
256.                                    </label>
257.                                </div>
```

```
                                <div className="form-selection-
  wrapper">
                                    <Field
                                        className="form-checkbox"
                                        type="checkbox"
                                        id="hobbySport"
                                        name="hobbySport"
                                    />
                                    <label
                                        className="form-selection-
  label"
                                        htmlFor="hobbySport"
                                    >
                                        Sport
                                    </label>
                                </div>

                                {/* Birthday */}
                                <label className="form-label"
  htmlFor="birthday">
                                    Birthday
                                </label>
                                <Field
                                    className="form-field"
                                    type="date"
                                    id="birthday"
                                    name="birthday"
                                />
                                <ErrorMessage
                                    className="form-error-message"
                                    name="birthday"
                                    component="div"
                                />

                                {/* Time */}
                                <label className="form-label"
  htmlFor="time">
                                    When can we call you?
                                </label>
                                <Field
                                    className="form-field"
                                    type="time"
                                    id="time"
                                    name="time"
                                />
                                <ErrorMessage
                                    className="form-error-message"
                                    name="time"
                                    component="div"
                                />

                                {/* Number */}
                                <label className="form-label"
  htmlFor="number">
                                    Pick a number from 0 to 100
                                </label>
                                <Field
                                    className="form-field"
```

```
                                    type="number"
                                    id="number"
                                    name="number"
                                    min="0"
                                    max="100"
                                />
                                <ErrorMessage
                                    className="form-error-message"
                                    name="number"
                                    component="div"
                                />

                                {/* Color */}
                                <label className="form-label"
  htmlFor="color">
                                    Your favorite color
                                </label>
                                <Field
                                    className="form-field"
                                    type="color"
                                    id="color"
                                    name="color"
                                />

                                {/* Rating */}
                                <label className="form-label"
  htmlFor="rating">
                                    Rate this form on a scale from 0
  to 10
                                </label>
                                <Field
                                    className="form-field"
                                    type="range"
                                    id="rating"
                                    name="rating"
                                    min="0"
                                    max="10"
                                />

                                {/* Season */}
                                <label className="form-label"
  htmlFor="season">
                                    Your favorite season
                                </label>
                                <Field
                                    className="form-field"
                                    as="select"
                                    id="season"
                                    name="season"
                                >
                                    <option
  value="summer">Summer</option>
                                    <option
  value="autumn">Autumn</option>
                                    <option
  value="winter">Winter</option>
                                    <option
  value="spring">Spring</option>
```

```
                                </Field>

                                {/* Comment */}
                                <label className="form-label"
  htmlFor="comment">
                                    Comment
                                </label>
                                <Field
                                    className="form-textarea"
                                    as="textarea"
                                    id="comment"
                                    name="comment"
                                    placeholder="Leave feedback
  about the form"
                                />

                                {/* Submit */}
                                <button
                                    className="form-button"
                                    type="submit"
                                    disabled={isSubmitting}
                                >
                                    Submit
                                </button>
                            </Form>
                        )}
                    </Formik>
                </div>
            </div>
        );
    }

    export default FormExample;
```

ДОДАТОК И

КОД ФАЙЛУ FORMEXAMPLE.JS (REACT HOOK FORM)

```
1. import React, { useRef } from "react";
2. import { useForm, Controller } from "react-hook-form";
3. import { yupResolver } from "@hookform/resolvers/yup";
4. import * as yup from "yup";
5.
6. function FormExample() {
7.
8.     const validationSchema = yup.object().shape({
9.         name: yup
10.             .string()
11.             .matches(/^[A-Za-z]+$/, "Name must contain only
   English letters")
12.             .required("Name is required"),
13.         email: yup
14.             .string()
15.             .email("Invalid email format")
16.             .required("Email is required"),
17.         phone: yup
18.             .string()
19.             .matches(
20.                 /^\+380\d{9}$/,
21.                 "Phone number must start with '+380' and contain
   12 digits"
22.             )
23.             .required("Phone number is required"),
24.         password: yup
25.             .string()
26.             .min(8, "Password must be at least 8 characters
   long")
27.             .matches(
28.                 /[a-z]/,
29.                 "Password must contain at least one lowercase
   letter"
30.             )
31.             .matches(
32.                 /[A-Z]/,
33.                 "Password must contain at least one uppercase
   letter"
34.             )
35.             .matches(/\d/, "Password must contain at least one
   number")
36.             .matches(
37.                 /[@$!%*?&#]/,
38.                 "Password must contain at least one special
   character"
39.             )
40.             .required("Password is required"),
41.         gender: yup.string().required("Gender is required"),
42.         birthday: yup
43.             .date()
44.             .required("Birthday is required")
45.             .test(
46.                 "age",
```

```
                    "You must be at least 13 years old and not older
  than 100 years (because it is impossible)",
                    (value) => {
                        if (!value) return false;
                        const today = new Date();
                        const birthDate = new Date(value);
                        const age = today.getFullYear() -
  birthDate.getFullYear();
                        const monthDiff = today.getMonth() -
  birthDate.getMonth();
                        if (
                            monthDiff < 0 ||
                            (monthDiff === 0 &&
                                today.getDate() <
  birthDate.getDate())
                        ) {
                            return age - 1 >= 13 && age - 1 <= 100;
                        }
                        return age >= 13 && age <= 100;
                    }
                ),
            time: yup.string().required("Time is required"),
            number: yup
                .number()
                .required("Number is required")
                .min(0, "Number must be at least 0")
                .max(100, "Number must be at most 100"),
        });

        const {
            register,
            handleSubmit,
            control,
            formState: { errors },
        } = useForm({
            resolver: yupResolver(validationSchema),
            defaultValues: {
                name: "",
                email: "",
                phone: "",
                password: "",
                gender: "",
                hobbyReading: false,
                hobbyDrawing: false,
                hobbySport: false,
                birthday: null,
                time: null,
                number: 0,
                color: "#EDBB36",
                rating: 10,
                season: "summer",
                comment: "",
            },
        });

        const onSubmit = (data) => {
            alert("Form submitted successfully!");
            console.log(data);
```

```
    };

    return (
        <div className="form-window">
            <div className="form-container">
                <div className="form-title">Example form</div>

                <form onSubmit={handleSubmit(onSubmit)}>
                    {/* Name */}
                    <label className="form-label"
  htmlFor="name">
                        Name
                    </label>
                    <input
                        className="form-field"
                        type="text"
                        id="name"
                        {...register("name")}
                        placeholder="James"
                    />
                    {errors.name && (
                        <div className="form-error-message">
                            {errors.name.message}
                        </div>
                    )}

                    {/* Email */}
                    <label className="form-label"
  htmlFor="email">
                        Email
                    </label>
                    <input
                        className="form-field"
                        type="text"
                        id="email"
                        {...register("email")}
                        placeholder="example@mail.com"
                    />
                    {errors.email && (
                        <div className="form-error-message">
                            {errors.email.message}
                        </div>
                    )}

                    {/* Phone */}
                    <label className="form-label"
  htmlFor="phone">
                        Phone number
                    </label>
                    <input
                        className="form-field"
                        type="text"
                        id="phone"
                        {...register("phone")}
                        placeholder="+380yyxxxxxxx"
                    />
                    {errors.phone && (
                        <div className="form-error-message">
```

```
                        {errors.phone.message}
                    </div>
                )}

                {/* Password */}
                <label className="form-label"
  htmlFor="password">
                    Password
                </label>
                <input
                    className="form-field"
                    type="password"
                    id="password"
                    {...register("password")}
                    placeholder="At least 8 characters"
                />
                {errors.password && (
                    <div className="form-error-message">
                        {errors.password.message}
                    </div>
                )}

                {/* Gender */}
                <div className="form-label">Gender</div>
                <div className="form-selection-wrapper">
                    <input
                        className="form-radio"
                        type="radio"
                        id="male"
                        value="male"
                        {...register("gender")}
                    />
                    <label className="form-selection-label"
  htmlFor="male">
                        Male
                    </label>
                </div>
                <div className="form-selection-wrapper">
                    <input
                        className="form-radio"
                        type="radio"
                        id="female"
                        value="female"
                        {...register("gender")}
                    />
                    <label
                        className="form-selection-label"
                        htmlFor="female"
                    >
                        Female
                    </label>
                </div>
                <div className="form-selection-wrapper">
                    <input
                        className="form-radio"
                        type="radio"
                        id="other"
                        value="other"
```

```
212.                             {...register("gender")}
213.                         />
214.                         <label className="form-selection-label"
    htmlFor="other">
215.                             Other
216.                         </label>
217.                     </div>
218.                     {errors.gender && (
219.                         <div className="form-error-message">
220.                             {errors.gender.message}
221.                         </div>
222.                     )}
223.
224.                     {/* Hobbies */}
225.                     <div className="form-label">Choose your
    hobbies</div>
226.                     <div className="form-selection-wrapper">
227.                         <input
228.                             className="form-checkbox"
229.                             type="checkbox"
230.                             id="hobbyReading"
231.                             {...register("hobbyReading")}
232.                         />
233.                         <label
234.                             className="form-selection-label"
235.                             htmlFor="hobbyReading"
236.                         >
237.                             Reading
238.                         </label>
239.                     </div>
240.                     <div className="form-selection-wrapper">
241.                         <input
242.                             className="form-checkbox"
243.                             type="checkbox"
244.                             id="hobbyDrawing"
245.                             {...register("hobbyDrawing")}
246.                         />
247.                         <label
248.                             className="form-selection-label"
249.                             htmlFor="hobbyDrawing"
250.                         >
251.                             Drawing
252.                         </label>
253.                     </div>
254.                     <div className="form-selection-wrapper">
255.                         <input
256.                             className="form-checkbox"
257.                             type="checkbox"
258.                             id="hobbySport"
259.                             {...register("hobbySport")}
260.                         />
261.                         <label
262.                             className="form-selection-label"
263.                             htmlFor="hobbySport"
264.                         >
265.                             Sport
266.                         </label>
267.                     </div>
```

```
268.
269.                     {/* Birthday */}
270.                     <label className="form-label"
  htmlFor="birthday">
271.                         Birthday
272.                     </label>
273.                     <input
274.                         className="form-field"
275.                         type="date"
276.                         id="birthday"
277.                         {...register("birthday")}
278.                     />
279.                     {errors.birthday && (
280.                         <div className="form-error-message">
281.                             {errors.birthday.message}
282.                         </div>
283.                     )}
284.
285.                     {/* Time */}
286.                     <label className="form-label"
  htmlFor="time">
287.                         When can we call you?
288.                     </label>
289.                     <input
290.                         className="form-field"
291.                         type="time"
292.                         id="time"
293.                         {...register("time")}
294.                     />
295.                     {errors.time && (
296.                         <div className="form-error-message">
297.                             {errors.time.message}
298.                         </div>
299.                     )}
300.
301.                     {/* Number */}
302.                     <label className="form-label"
  htmlFor="number">
303.                         Pick a number from 0 to 100
304.                     </label>
305.                     <input
306.                         className="form-field"
307.                         type="number"
308.                         id="number"
309.                         {...register("number")}
310.                         min="0"
311.                         max="100"
312.                     />
313.                     {errors.number && (
314.                         <div className="form-error-message">
315.                             {errors.number.message}
316.                         </div>
317.                     )}
318.
319.                     {/* Color */}
320.                     <label className="form-label"
  htmlFor="color">
321.                         Your favorite color
```

```
                    </label>
                    <Controller
                        name="color"
                        control={control}
                        render={({ field }) => (
                            <input
                                className="form-field"
                                type="color"
                                id="color"
                                {...field}
                            />
                        )}
                    />

                    {/* Rating */}
                    <label className="form-label"
  htmlFor="rating">
                        Rate this form on a scale from 0 to 10
                    </label>
                    <input
                        className="form-field"
                        type="range"
                        id="rating"
                        {...register("rating")}
                        min="0"
                        max="10"
                    />

                    {/* Season */}
                    <label className="form-label"
  htmlFor="season">
                        Your favorite season
                    </label>
                    <select
                        className="form-field"
                        id="season"
                        {...register("season")}
                    >
                        <option value="summer">Summer</option>
                        <option value="autumn">Autumn</option>
                        <option value="winter">Winter</option>
                        <option value="spring">Spring</option>
                    </select>

                    {/* Comment */}
                    <label className="form-label"
  htmlFor="comment">
                        Comment
                    </label>
                    <textarea
                        className="form-textarea"
                        id="comment"
                        {...register("comment")}
                        placeholder="Leave feedback about the
  form"
                    />

                    {/* Submit */}
```

```
376.                       <button className="form-button"
   type="submit">
377.                           Submit
378.                       </button>
379.                   </form>
380.               </div>
381.           </div>
382.       );
383.   }
384.
385.   export default FormExample;
```

ДОДАТОК К

КОД ФАЙЛУ FORMEXAMPLE.JS (FINAL FORM)

```
1. import React, { useRef } from "react";
2. import { Form, Field } from "react-final-form";
3. import * as yup from "yup";
4.
5. function FormExample() {
6.
7.     const validationSchema = yup.object().shape({
8.         name: yup
9.             .string()
10.                .matches(/^[A-Za-z]+$/, "Name must contain only
   English letters")
11.                .required("Name is required"),
12.            email: yup
13.                .string()
14.                .email("Invalid email format")
15.                .required("Email is required"),
16.            phone: yup
17.                .string()
18.                .matches(
19.                    /^\+380\d{9}$/,
20.                    "Phone number must start with '+380' and contain
   12 digits"
21.                )
22.                .required("Phone number is required"),
23.            password: yup
24.                .string()
25.                .min(8, "Password must be at least 8 characters
   long")
26.                .matches(
27.                    /[a-z]/,
28.                    "Password must contain at least one lowercase
   letter"
29.                )
30.                .matches(
31.                    /[A-Z]/,
32.                    "Password must contain at least one uppercase
   letter"
33.                )
34.                .matches(/\d/, "Password must contain at least one
   number")
35.                .matches(
36.                    /[@$!%*?&#]/,
37.                    "Password must contain at least one special
   character"
38.                )
39.                .required("Password is required"),
40.            gender: yup.string().required("Gender is required"),
41.            birthday: yup
42.                .date()
43.                .required("Birthday is required")
44.                .test(
45.                    "age",
```

```
                    "You must be at least 13 years old and not older
  than 100 years (because it is impossible)",
                    (value) => {
                        if (!value) return false;
                        const today = new Date();
                        const birthDate = new Date(value);
                        const age = today.getFullYear() -
  birthDate.getFullYear();
                        const monthDiff = today.getMonth() -
  birthDate.getMonth();
                        if (
                            monthDiff < 0 ||
                            (monthDiff === 0 &&
                                today.getDate() <
  birthDate.getDate())
                        ) {
                            return age - 1 >= 13 && age - 1 <= 100;
                        }
                        return age >= 13 && age <= 100;
                    }
                ),
            time: yup.string().required("Time is required"),
            number: yup
                .number()
                .required("Number is required")
                .min(0, "Number must be at least 0")
                .max(100, "Number must be at most 100"),
        });

        const validate = async (values) => {
            try {
                await validationSchema.validate(values, {
  abortEarly: false });
            } catch (err) {
                if (err instanceof yup.ValidationError) {
                    return err.inner.reduce((errors, error) => {
                        errors[error.path] = error.message;
                        return errors;
                    }, {});
                }
            }
        };

        const onSubmit = async (values, form) => {
            alert("Form submitted successfully!");
            form.reset();
        };

        return (
            <div className="form-window">
                <div className="form-container">
                    <div className="form-title">Example form</div>
                    <Form
                        onSubmit={onSubmit}
                        validate={validate}
                        initialValues={{
                            name: "",
                            email: "",
```

```
                        phone: "",
                        password: "",
                        gender: "",
                        hobbyReading: false,
                        hobbyDrawing: false,
                        hobbySport: false,
                        birthday: null,
                        time: null,
                        number: 0,
                        color: "#EDBB36",
                        rating: 10,
                        season: "summer",
                        comment: "",
                    }}
                    render={({ handleSubmit, submitError }) => (
                        <form onSubmit={handleSubmit}>
                            {submitError && (
                                <div className="form-error-
  message">
                                    {submitError}
                                </div>
                            )}
                            {/* Name */}
                            <Field name="name">
                                {({ input, meta }) => (
                                    <>
                                        <label
                                            className="form-
  label"
                                            htmlFor="name"
                                        >
                                            Name
                                        </label>
                                        <input
                                            {...input}
                                            className="form-
  field"
                                            id="name"
                                            placeholder="James"
                                        />
                                        {meta.touched &&
  meta.error && (
                                            <span
  className="form-error-message">
                                                {meta.error}
                                            </span>
                                        )}
                                    </>
                                )}
                            </Field>
                            {/* Email */}
                            <Field name="email">
                                {({ input, meta }) => (
                                    <>
                                        <label
                                            className="form-
  label"
                                            htmlFor="email"
```

```
                                        >
                                            Email
                                        </label>
                                        <input
                                            {...input}
                                            className="form-
  field"
                                            id="email"

  placeholder="example@mail.com"
                                        />
                                        {meta.touched &&
  meta.error && (
                                            <span
  className="form-error-message">
                                                {meta.error}
                                            </span>
                                        )}
                                    </>
                                )}
                            </Field>
                            {/* Phone */}
                            <Field name="phone">
                                {({ input, meta }) => (
                                    <>
                                        <label
                                            className="form-
  label"
                                            htmlFor="phone"
                                        >
                                            Phone number
                                        </label>
                                        <input
                                            {...input}
                                            className="form-
  field"
                                            id="phone"

  placeholder="+380yyxxxxxxx"
                                        />
                                        {meta.touched &&
  meta.error && (
                                            <span
  className="form-error-message">
                                                {meta.error}
                                            </span>
                                        )}
                                    </>
                                )}
                            </Field>
                            {/* Password */}
                            <Field name="password">
                                {({ input, meta }) => (
                                    <>
                                        <label
                                            className="form-
  label"
                                            htmlFor="password"
```

```
                                        >
                                            Password
                                        </label>
                                        <input
                                            {...input}
                                            className="form-
  field"
                                            id="password"
                                            type="password"
                                            placeholder="At
  least 8 characters"
                                        />
                                        {meta.touched &&
  meta.error && (
                                            <span
  className="form-error-message">
                                                {meta.error}
                                            </span>
                                        )}
                                    </>
                                )}
                            </Field>
                            {/* Gender */}
                            <div className="form-
  label">Gender</div>
                            <Field name="gender" type="radio"
  value="male">
                                {({ input }) => (
                                    <div className="form-
  selection-wrapper">
                                        <input
                                            {...input}
                                            className="form-
  radio"
                                            id="male"
                                        />
                                        <label
                                            className="form-
  selection-label"
                                            htmlFor="male"
                                        >
                                            Male
                                        </label>
                                    </div>
                                )}
                            </Field>
                            <Field name="gender" type="radio"
  value="female">
                                {({ input }) => (
                                    <div className="form-
  selection-wrapper">
                                        <input
                                            {...input}
                                            className="form-
  radio"
                                            id="female"
                                        />
                                        <label
```

```
                                    className="form-
                                    htmlFor="female"
                                >
                                    Female
                                </label>
                            </div>
                        )}
                    </Field>
                    <Field name="gender" type="radio"
                        {({ input }) => (
                            <div className="form-
                                <input
                                    {...input}
                                    className="form-
                                    id="other"
                                />
                                <label
                                    className="form-
                                    htmlFor="other"
                                >
                                    Other
                                </label>
                            </div>
                        )}
                    </Field>
                    {/* Gender error display */}
                    <Field name="gender">
                        {({ meta }) =>
                            meta.touched && meta.error ?
                                <span className="form-
                                    {meta.error}
                                </span>
                            ) : null
                        }
                    </Field>
                    {/* Hobbies */}
                    <div className="form-label">
                        Choose your hobbies
                    </div>
                    <Field name="hobbyReading"
                        {({ input }) => (
                            <div className="form-
                                <input
                                    {...input}
                                    className="form-
                                    id="hobbyReading"
                                />
                                <label
```

```
293.                                                    className="form-
   selection-label"
294.
   htmlFor="hobbyReading"
295.                                                >
296.                                                    Reading
297.                                                </label>
298.                                            </div>
299.                                        )}
300.                                    </Field>
301.                                    <Field name="hobbyDrawing"
   type="checkbox">
302.                                        {({ input }) => (
303.                                            <div className="form-
   selection-wrapper">
304.                                                <input
305.                                                    {...input}
306.                                                    className="form-
   checkbox"
307.                                                    id="hobbyDrawing"
308.                                                />
309.                                                <label
310.                                                    className="form-
   selection-label"
311.
   htmlFor="hobbyDrawing"
312.                                                >
313.                                                    Drawing
314.                                                </label>
315.                                            </div>
316.                                        )}
317.                                    </Field>
318.                                    <Field name="hobbySport"
   type="checkbox">
319.                                        {({ input }) => (
320.                                            <div className="form-
   selection-wrapper">
321.                                                <input
322.                                                    {...input}
323.                                                    className="form-
   checkbox"
324.                                                    id="hobbySport"
325.                                                />
326.                                                <label
327.                                                    className="form-
   selection-label"
328.                                                    htmlFor="hobbySport"
329.                                                >
330.                                                    Sport
331.                                                </label>
332.                                            </div>
333.                                        )}
334.                                    </Field>
335.                                    {/* Birthday */}
336.                                    <Field name="birthday">
337.                                        {({ input, meta }) => (
338.                                            <>
339.                                                <label
```

```
                                        className="form-
  label"
                                        htmlFor="birthday"
                                    >
                                        Birthday
                                    </label>
                                    <input
                                        {...input}
                                        className="form-
  field"
                                        id="birthday"
                                        type="date"
                                    />
                                    {meta.touched &&
  meta.error && (
                                        <span
  className="form-error-message">
                                            {meta.error}
                                        </span>
                                    )}
                                </>
                            )}
                        </Field>
                        {/* Time */}
                        <Field name="time">
                            {({ input, meta }) => (
                                <>
                                    <label
                                        className="form-
  label"
                                        htmlFor="time"
                                    >
                                        When can we call
  you?
                                    </label>
                                    <input
                                        {...input}
                                        className="form-
  field"
                                        id="time"
                                        type="time"
                                    />
                                    {meta.touched &&
  meta.error && (
                                        <span
  className="form-error-message">
                                            {meta.error}
                                        </span>
                                    )}
                                </>
                            )}
                        </Field>
                        {/* Number */}
                        <Field name="number">
                            {({ input, meta }) => (
                                <>
                                    <label
```

```
                                        className="form-
  label"
                                        htmlFor="number"
                                    >
                                        Pick a number from 0
  to 100
                                    </label>
                                    <input
                                        {...input}
                                        className="form-
  field"
                                        id="number"
                                        type="number"
                                        min="0"
                                        max="100"
                                    />
                                    {meta.touched &&
  meta.error && (
                                        <span
  className="form-error-message">
                                            {meta.error}
                                        </span>
                                    )}
                                </>
                            )}
                        </Field>
                        {/* Color */}
                        <Field name="color">
                            {({ input }) => (
                                <>
                                    <label
                                        className="form-
  label"
                                        htmlFor="color"
                                    >
                                        Your favorite color
                                    </label>
                                    <input
                                        {...input}
                                        className="form-
  field"
                                        id="color"
                                        type="color"
                                    />
                                </>
                            )}
                        </Field>
                        {/* Rating */}
                        <Field name="rating">
                            {({ input }) => (
                                <>
                                    <label
                                        className="form-
  label"
                                        htmlFor="rating"
                                    >
                                        Rate this form on a
  scale from 0 to
```

```
                                10
                            </label>
                            <input
                                {...input}
                                className="form-
field"
                                id="rating"
                                type="range"
                                min="0"
                                max="10"
                            />
                        </>
                    )}
                </Field>
                {/* Season */}
                <Field name="season">
                    {({ input }) => (
                        <>
                            <label
                                className="form-
label"
                                htmlFor="season"
                            >
                                Your favorite season
                            </label>
                            <select
                                {...input}
                                className="form-
field"
                                id="season"
                            >
                                <option
value="summer">
                                    Summer
                                </option>
                                <option
value="autumn">
                                    Autumn
                                </option>
                                <option
value="winter">
                                    Winter
                                </option>
                                <option
value="spring">
                                    Spring
                                </option>
                            </select>
                        </>
                    )}
                </Field>
                {/* Comment */}
                <Field name="comment">
                    {({ input }) => (
                        <>
                            <label
                                className="form-
label"
```

```
487.                                                htmlFor="comment"
488.                                            >
489.                                                Comment
490.                                            </label>
491.                                            <textarea
492.                                                {...input}
493.                                                className="form-
   textarea"
494.                                                id="comment"
495.                                                placeholder="Leave
   feedback about the form"
496.                                            />
497.                                        </>
498.                                    )}
499.                                </Field>
500.                                <button className="form-button"
   type="submit">
501.                                    Submit
502.                                </button>
503.                            </form>
504.                        )}
505.                    />
506.                </div>
507.            </div>
508.        );
509.  }
510.
511.  export default FormExample;
```

ДОДАТОК Л

КОД ФАЙЛУ FORMEXAMPLE.JS (REDUX FORM)

```
1. import React from "react";
2. import { Field, reduxForm } from "redux-form";
3. import * as yup from "yup";
4.
5. const validationSchema = yup.object().shape({
6.     name: yup
7.         .string()
8.         .matches(/^[A-Za-z]+$/, "Name must contain only English
   letters")
9.         .required("Name is required"),
10.       email: yup
11.           .string()
12.           .email("Invalid email format")
13.           .required("Email is required"),
14.       phone: yup
15.           .string()
16.           .matches(
17.               /^\+380\d{9}$/,
18.               "Phone number must start with '+380' and contain 12
   digits"
19.           )
20.           .required("Phone number is required"),
21.       password: yup
22.           .string()
23.           .min(8, "Password must be at least 8 characters long")
24.           .matches(/[a-z]/, "Password must contain at least one
   lowercase letter")
25.           .matches(/[A-Z]/, "Password must contain at least one
   uppercase letter")
26.           .matches(/\d/, "Password must contain at least one
   number")
27.           .matches(
28.               /[@$!%*?&#]/,
29.               "Password must contain at least one special
   character"
30.           )
31.           .required("Password is required"),
32.       gender: yup.string().required("Gender is required"),
33.       birthday: yup
34.           .date()
35.           .required("Birthday is required")
36.           .test(
37.               "age",
38.               "You must be at least 13 years old and not older
   than 100 years",
39.               (value) => {
40.                   if (!value) return false;
41.                   const today = new Date();
42.                   const birthDate = new Date(value);
43.                   const age = today.getFullYear() -
   birthDate.getFullYear();
44.                   const monthDiff = today.getMonth() -
   birthDate.getMonth();
```

```
                    if (
                        monthDiff < 0 ||
                        (monthDiff === 0 && today.getDate() <
  birthDate.getDate())
                    ) {
                        return age - 1 >= 13 && age - 1 <= 100;
                    }
                    return age >= 13 && age <= 100;
                }
            ),
        time: yup.string().required("Time is required"),
        number: yup
            .number()
            .required("Number is required")
            .min(0, "Number must be at least 0")
            .max(100, "Number must be at most 100"),
    });

    const validate = (values) => {
        const errors = {};
        try {
            validationSchema.validateSync(values, { abortEarly:
  false });
        } catch (err) {
            if (err.inner) {
                err.inner.forEach((validationError) => {
                    errors[validationError.path] =
  validationError.message;
                });
            }
        }
        return errors;
    };

    const renderField = ({
        input,
        label,
        type,
        meta: { touched, error },
        placeholder,
        min,
        max,
    }) => (
        <div>
            <label className="form-label">{label}</label>
            <input
                {...input}
                type={type}
                className="form-field"
                placeholder={placeholder}
                min={min}
                max={max}
            />
            {touched && error && (
                <span className="form-error-message">{error}</span>
            )}
        </div>
    );
```

```
100.
101.  const renderRadioGroup = ({
102.      input,
103.      options,
104.      label,
105.      meta: { touched, error },
106.  }) => (
107.      <div>
108.          <div className="form-label">{label}</div>
109.          {options.map((option) => (
110.              <div className="form-selection-wrapper"
    key={option.value}>
111.                  <input
112.                      {...input}
113.                      type="radio"
114.                      value={option.value}
115.                      checked={input.value === option.value}
116.                      className="form-radio"
117.                  />
118.                  <label className="form-selection-
    label">{option.label}</label>
119.              </div>
120.          ))}
121.          {touched && error && (
122.              <span className="form-error-message">{error}</span>
123.          )}
124.      </div>
125.  );
126.
127.  const renderCheckboxGroup = ({ input, options, label }) => (
128.      <div>
129.          <div className="form-label">{label}</div>
130.          {options.map((option) => (
131.              <div className="form-selection-wrapper"
    key={option.name}>
132.                  <input
133.                      type="checkbox"
134.                      name={option.name}
135.                      checked={input.value.includes(option.name)}
136.                      onChange={(event) => {
137.                          const newValue = [...input.value];
138.                          if (event.target.checked) {
139.                              newValue.push(option.name);
140.                          } else {
141.
    newValue.splice(newValue.indexOf(option.name), 1);
142.                          }
143.                          input.onChange(newValue);
144.                      }}
145.                      className="form-checkbox"
146.                  />
147.                  <label className="form-selection-
    label">{option.label}</label>
148.              </div>
149.          ))}
150.      </div>
151.  );
152.
```

```
153. const renderSelect = ({ input, label, options, meta: { touched,
   error } }) => (
154.     <div>
155.         <label className="form-label">{label}</label>
156.         <select {...input} className="form-field">
157.             {options.map((option) => (
158.                 <option key={option.value} value={option.value}>
159.                     {option.label}
160.                 </option>
161.             ))}
162.         </select>
163.         {touched && error && (
164.             <span className="form-error-message">{error}</span>
165.         )}
166.     </div>
167. );
168.
169. const renderTextarea = ({
170.     input,
171.     label,
172.     meta: { touched, error },
173.     placeholder,
174. }) => (
175.     <div>
176.         <label className="form-label">{label}</label>
177.         <textarea
178.             {...input}
179.             className="form-textarea"
180.             placeholder={placeholder}
181.         />
182.         {touched && error && (
183.             <span className="form-error-message">{error}</span>
184.         )}
185.     </div>
186. );
187.
188. const FormExample = ({ handleSubmit, initialize, valid }) => {
189.     React.useEffect(() => {
190.         initialize({
191.             number: 0,
192.             color: "#EDBB36",
193.             rating: 10,
194.             season: "summer",
195.             hobbies: [],
196.         });
197.     }, [initialize]);
198.
199.     const onSubmit = (values) => {
200.         if (!valid) {
201.             return;
202.         }
203.         alert("Form submitted successfully!");
204.         console.log(values);
205.     };
206.
207.     return (
208.         <form className="form-window"
   onSubmit={handleSubmit(onSubmit)}>
```

```
            <div className="form-container">
                <div className="form-title">Example form</div>

                <Field
                    name="name"
                    type="text"
                    component={renderField}
                    label="Name"
                    placeholder="James"
                />
                <Field
                    name="email"
                    type="text"
                    component={renderField}
                    label="Email"
                    placeholder="example@mail.com"
                />
                <Field
                    name="phone"
                    type="text"
                    component={renderField}
                    label="Phone number"
                    placeholder="+380yyxxxxxxx"
                />
                <Field
                    name="password"
                    type="password"
                    component={renderField}
                    label="Password"
                    placeholder="At least 8 characters"
                />

                <Field
                    name="gender"
                    component={renderRadioGroup}
                    label="Gender"
                    options={[
                        { value: "male", label: "Male" },
                        { value: "female", label: "Female" },
                        { value: "other", label: "Other" },
                    ]}
                />

                <Field
                    name="hobbies"
                    component={renderCheckboxGroup}
                    label="Choose your hobbies"
                    options={[
                        { name: "hobbyReading", label: "Reading"
  },
                        { name: "hobbyDrawing", label: "Drawing"
  },
                        { name: "hobbySport", label: "Sport" },
                    ]}
                />

                <Field
                    name="birthday"
```

```
                    type="date"
                    component={renderField}
                    label="Birthday"
                />
                <Field
                    name="time"
                    type="time"
                    component={renderField}
                    label="When can we call you?"
                />
                <Field
                    name="number"
                    type="number"
                    component={renderField}
                    label="Pick a number from 0 to 100"
                    min={0}
                    max={100}
                />
                <Field
                    name="color"
                    type="color"
                    component={renderField}
                    label="Your favorite color"
                />
                <Field
                    name="rating"
                    type="range"
                    component={renderField}
                    label="Rate this form on a scale from 0 to
  10"
                    min={0}
                    max={10}
                />

                <Field
                    name="season"
                    component={renderSelect}
                    label="Your favorite season"
                    options={[
                        { value: "summer", label: "Summer" },
                        { value: "autumn", label: "Autumn" },
                        { value: "winter", label: "Winter" },
                        { value: "spring", label: "Spring" },
                    ]}
                />

                <Field
                    name="comment"
                    component={renderTextarea}
                    label="Comment"
                    placeholder="Leave feedback about the form"
                />

                <button className="form-button" type="submit">
                    Submit
                </button>
            </div>
        </form>
```

```
322.     );
323. };
324.
325. export default reduxForm({
326.     form: "exampleForm",
327.     validate,
328.     destroyOnUnmount: false,
329.     enableReinitialize: false,
330. })(FormExample);
```

ДОДАТОК М

ІНТЕРФЕЙС РОЗРОБЛЕНОЇ ПРОГРАМИ