

ДОДАТОК А

РЕЗУЛЬТАТ ПРОХОДЖЕННЯ СИСТЕМИ ПЕРЕВІРКИ ДОБРОЧЕСНОСТІ



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
Науковий керівник / Експерт
Дудник Олександр Олегович 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.



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		1
Інтервали		0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)		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 5/22/2025 Lutsk National Technical University course papers (Lutsk National Technical University course papers)	13 0.09 %

ДОДАТОК Б СЛАЙДИ ПРЕЗЕНТАЦІЇ

Дослідження ефективності різних підходів до розробки форм у React

Дудник О. О., ІПЗм-23-2
Науковий керівник: доц. каф. ПІ, Каук В. І.

Дослідження

Актуальність та стан розвитку галузі:

- React займає провідне положення серед JavaScript-фреймворків у сфері веб-розробки;
- є стійка тенденція до зростання популярності React в професійному середовищі;
- існує значна кількість бібліотек для управління формами, що актуалізує проблему вибору найбільш ефективного рішення;
- відсутні систематичні емпіричні дослідження продуктивності та ефективності різних підходів до роботи з формами.

Визначення напрямку дослідження:

- порівняльний аналіз ефективності різних підходів до створення форм у React;
- виявлення найбільш ефективного підходу до створення форм у React;
- надання чітких рекомендацій щодо вибору підходу до розробки форм у React.

Об'єкт дослідження:

Розробка форм у React, інструменти та підходи до їх створення.

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.

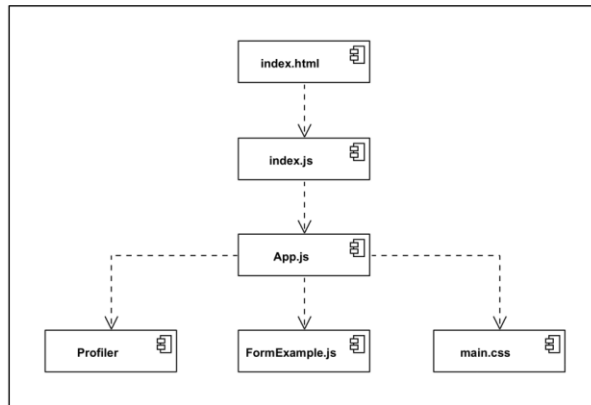


Архітектура системи для проведення експериментального дослідження

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

Example form

Name:

Email:

Phone number:

Password:

Gender: Male Female Other

Choose your hobbies: Reading Drawing Sport

Birthdate:

When can we call you?

Pick a number from 0 to 100:

Your favorite color:

Rate this form on a scale from 0 to 10:

Your favorite season:

Comment:



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 рендерів	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	



13

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.



14

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



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



Публікація результатів

MIT@AIS Conference 2025

ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ РІЗНИХ ПІДХОДІВ ДО РОЗРОБКИ ФОРМ У REACT

Дудник О. О., Кувш В. І.

1. Харківський національний університет радіоелектроніки, пр. Леніна, 14 м. Харків, 61104, Україна
oleksandr.dudnyk@nure.ua, vku@nure.ua

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

АНОТАЦІЯ
У тексті представлено результати дослідження ефективності різних підходів до розробки форм у React-застосунках. Протягом порівняльної аналізи впроваджені «чистий» React та популярні бібліотеки Formik, React Hook Form, Final Form і Redux Form. Оцінка проводилася за кількісними показниками: кількість рядків коду, час першого рендерингу, кількість рендерів, об'єм споживаних пам'яті, час виклику DTP та розмір бібліотеки у фінальному будівлі. Встановлено, що найкращу загальну продуктивність демонструє бібліотека React Hook Form. Redux Form, попри компактність коду, показує гірші результати за більшістю метрик.

ПЕРЕГЛЯДОВА
React є мовою з найпопулярнішим бібліотекою для розробки інтерфейсів користувача у веб-середовищі серед інших частин розробки [1]. Форми є важливою частиною сучасних веб-застосунків, проте на цей час відсутня комплексна дослідження, які б порівнювали різні підходи до створення форм у React на об'єднаному методичному рівні. 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'єднана 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 встановлено метри 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 Sloppy DetTools та React Profiler [2]. Порівняння проводилося за наступними метриками: кількість рядків коду, час першого рендерингу, кількість рендерів, мінімальний та максимальний об'єм споживаної пам'яті, час виклику DTP, розмір бібліотеки та її залежностей у фінальному будівлі.

*Corresponding author (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

Метрика	«Чистий» React	Formik	React Hook Form	Final Form	Redux Form
Кількість рядків коду	401	358	331	418	312
Час першого рендерингу (мс)	7.4	12.4	8.9	14.8	16.5
Кількість рендерів	137	173	4	12	183
Мінімальний об'єм споживаної пам'яті (KB)	3.7	4.2	4.1	4.2	5.7
Максимальний об'єм споживаної пам'яті (KB)	11.3	12.3	9.4	21.9	13.3
Час виклику DTP (мс)	32	40	32	40	32
Розмір бібліотеки (та залежностей) у будівлі (KB)	0	44.8	31.3	26	99.1

Найвищим об'ємом коду потребує Redux Form (312 рядків), тоді як найменшим – Final Form (418 рядків). Час першого рендерингу варіюється від 7,6 мс у «чистого» React до 16,5 мс у Redux Form. React Hook Form продемонструвала лише 4 рендеринги проти 183 у Redux Form. При аналізі використання пам'яті «чистий» React показав мінімальне споживання (3,7 KB), тоді як Final Form максимізує (21,9 KB). За метрикою DTP «чистий» React, React Hook Form та Redux Form показали результати у 32 мс, тоді як Formik і Final Form були повільнішими (40 мс). Щодо розміру фінального будівлі, Redux Form займає найбільше (99,1 KB), а Final Form – найменше серед бібліотек (26 KB); «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 проекту. Для великих проєктів з обмеженими ресурсами або критичними вимогами до швидкості завантаження доцільно використовувати «чистий» React, який забезпечує найвищий час першого рендерингу та не додає ваги до будівлі. React Hook Form можна рекомендацію як найбільш збалансоване рішення для більшості проєктів, оскільки на бібліотеку демонструє найвищу продуктивність за кількістю рендерів та споживаним пам'яті, забезпечує швидкий виклик інтерфейсу та помірний розмір будівлі. Final Form та Formik представляють собою компромісні рішення з середнім показником ефективності, тоді як Redux Form, попри компактність коду, демонструє гірші результати за більшістю метрик, що робить його дослідження перш за все, проєктом, де він використовується Redux.

ДЖЕРЕЛА

- V. Kompeira, D. Prastha, P. Ghah, R. Patta, React: A detailed survey, Indonesian Journal of Electrical Engineering and Computer Science 26 (2022) 1710. doi:10.1109/ijees.2024.1071710-1717.
- S. Moudal, Enhancing React Application Performance: Proven Strategies and Best Practices, International Research Journal of Engineering and Technology (IJRET) 11 (2024) 309-312.

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

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 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 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 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. 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 позначки	
	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-
178.         rotate(356deg) brightness(98%) contrast(92%);
179. }
```

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,
28.                name).validate(value);
29.            setFieldErrors((prev) => ({
30.                ...prev,
31.                [name]: undefined,
32.            }));
33.        } catch (err) {
34.            setFieldErrors((prev) => ({
35.                ...prev,
36.                [name]: err.message,
37.            }));
38.        }
39.    };
40.
41.    const handleChange = (event) => {
42.        const { name, value } = event.target;
43.        setFormData((prev) => ({
44.            ...prev,
45.            [name]: value,
46.        }));
47.        validateField(name, value);
48.    };
49.
50.    const handleCheckboxChange = (event) => {
51.        const { name, checked } = event.target;
52.        setFormData((prev) => ({
53.            ...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();

```

```

102.         const monthDiff = today.getMonth() -
           birthDate.getMonth();
103.         if (
104.             monthDiff < 0 ||
105.             (monthDiff === 0 &&
106.              today.getDate() <
           birthDate.getDate())
107.         ) {
108.             return age - 1 >= 13 && age - 1 <= 100;
109.         }
110.         return age >= 13 && age <= 100;
111.     }
112.     ),
113.     time: yup.string().required("Time is required"),
114.     number: yup
115.         .number()
116.         .required("Number is required")
117.         .min(0, "Number must be at least 0")
118.         .max(100, "Number must be at most 100"),
119. });
120.
121.     const handleSubmit = async () => {
122.         try {
123.             await validationSchema.validate(formData, {
           abortEarly: false });
124.             alert("Form submitted successfully!");
125.             setFieldErrors({});
126.         } catch (err) {
127.             if (err instanceof yup.ValidationError) {
128.                 const errors = {};
129.                 err.inner.forEach((error) => {
130.                     errors[error.path] = error.message;
131.                 });
132.                 setFieldErrors(errors);
133.             }
134.         }
135.     };
136.
137.     return (
138.         <div className="form-window">
139.             <div className="form-container">
140.                 <div className="form-title">Example form</div>
141.
142.                 {/* Name */}
143.                 <label className="form-label" htmlFor="name">
144.                     Name
145.                 </label>
146.                 <input
147.                     className="form-field"
148.                     type="text"
149.                     id="name"
150.                     name="name"
151.                     value={formData.name}
152.                     onChange={handleChange}
153.                     placeholder="James"
154.                 />
155.                 {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">

```

```

212.         {fieldErrors.password}
213.     </div>
214. }}
215.
216.     { /* Gender */ }
217.     <div className="form-label">Gender</div>
218.
219.     <div className="form-selection-wrapper">
220.         <input
221.             className="form-radio"
222.             type="radio"
223.             id="male"
224.             name="gender"
225.             value="male"
226.             checked={formData.gender === "male"}
227.             onChange={handleChange}
228.         />
229.         <label className="form-selection-label"
230.             htmlFor="male">
231.             Male
232.         </label>
233.     </div>
234.
235.     <div className="form-selection-wrapper">
236.         <input
237.             className="form-radio"
238.             type="radio"
239.             id="female"
240.             name="gender"
241.             value="female"
242.             checked={formData.gender === "female"}
243.             onChange={handleChange}
244.         />
245.         <label className="form-selection-label"
246.             htmlFor="female">
247.             Female
248.         </label>
249.     </div>
250.
251.     <div className="form-selection-wrapper">
252.         <input
253.             className="form-radio"
254.             type="radio"
255.             id="other"
256.             name="gender"
257.             value="other"
258.             checked={formData.gender === "other"}
259.             onChange={handleChange}
260.         />
261.         <label className="form-selection-label"
262.             htmlFor="other">
263.             Other
264.         </label>
265.     </div>
266.     {fieldErrors.gender && (
267.         <div className="form-error-message">
268.             {fieldErrors.gender}
269.         </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 */}

```

```

324.         <label className="form-label"
htmlFor="birthday">
325.             Birthday
326.         </label>
327.         <input
328.             className="form-field"
329.             type="date"
330.             id="birthday"
331.             name="birthday"
332.             value={formData.birthday}
333.             onChange={handleChange}
334.         />
335.         {fieldErrors.birthday && (
336.             <div className="form-error-message">
337.                 {fieldErrors.birthday}
338.             </div>
339.         )}
340.
341.         {/* Time */}
342.         <label className="form-label" htmlFor="time">
343.             When can we call you?
344.         </label>
345.         <input
346.             className="form-field"
347.             type="time"
348.             id="time"
349.             name="time"
350.             value={formData.time}
351.             onChange={handleChange}
352.         />
353.         {fieldErrors.time && (
354.             <div className="form-error-
message">{fieldErrors.time}</div>
355.         )}
356.
357.         {/* Number */}
358.         <label className="form-label" htmlFor="number">
359.             Pick a number from 0 to 100
360.         </label>
361.         <input
362.             className="form-field"
363.             type="number"
364.             id="number"
365.             name="number"
366.             value={formData.number}
367.             onChange={handleChange}
368.             min="0"
369.             max="100"
370.         />
371.         {fieldErrors.number && (
372.             <div className="form-error-message">
373.                 {fieldErrors.number}
374.             </div>
375.         )}
376.
377.         {/* Color */}
378.         <label className="form-label" htmlFor="color">
379.             Your favorite color

```

```

380.     </label>
381.     <input
382.         className="form-field"
383.         type="color"
384.         id="color"
385.         name="color"
386.         value={formData.color}
387.         onChange={handleChange}
388.     />
389.
390.     {/ * Rating */}
391.     <label className="form-label" htmlFor="rating">
392.         Rate this form on a scale from 0 to 10
393.     </label>
394.     <input
395.         className="form-field"
396.         type="range"
397.         id="rating"
398.         name="rating"
399.         value={formData.rating}
400.         onChange={handleChange}
401.         min="0"
402.         max="10"
403.     />
404.
405.     {/ * Season */}
406.     <label className="form-label" htmlFor="season">
407.         Your favorite season
408.     </label>
409.     <select
410.         className="form-field"
411.         id="season"
412.         name="season"
413.         value={formData.season}
414.         onChange={handleChange}
415.     >
416.         <option value="summer">Summer</option>
417.         <option value="autumn">Autumn</option>
418.         <option value="winter">Winter</option>
419.         <option value="spring">Spring</option>
420.     </select>
421.
422.     {/ * Comment */}
423.     <label className="form-label" htmlFor="comment">
424.         Comment
425.     </label>
426.     <textarea
427.         className="form-textarea"
428.         id="comment"
429.         name="comment"
430.         value={formData.comment}
431.         onChange={handleChange}
432.         placeholder="Leave feedback about the form"
433.     />
434.
435.     {/ * Submit */}
436.     <button className="form-button"
         onClick={handleSubmit}>

```

```
437.             Submit
438.             </button>
439.         </div>
440.     </div>
441.     );
442. }
443.
444. 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]/,
```

```

50.             "Password must contain at least one uppercase
   letter"
51.         )
52.         .matches(/\d/, "Password must contain at least one
   number")
53.         .matches(
54.             /[@$!%*?&#]/,
55.             "Password must contain at least one special
   character"
56.         )
57.         .required("Password is required"),
58.     gender: yup.string().required("Gender is required"),
59.     birthday: yup
60.         .date()
61.         .required("Birthday is required")
62.         .test(
63.             "age",
64.             "You must be at least 13 years old and not older
   than 100 years (because it is impossible)",
65.             (value) => {
66.                 if (!value) return false;
67.                 const today = new Date();
68.                 const birthDate = new Date(value);
69.                 const age = today.getFullYear() -
   birthDate.getFullYear();
70.                 const monthDiff = today.getMonth() -
   birthDate.getMonth();
71.                 if (
72.                     monthDiff < 0 ||
73.                     (monthDiff === 0 &&
74.                     today.getDate() <
   birthDate.getDate())
75.                 ) {
76.                     return age - 1 >= 13 && age - 1 <= 100;
77.                 }
78.                 return age >= 13 && age <= 100;
79.             }
80.         ),
81.     time: yup.string().required("Time is required"),
82.     number: yup
83.         .number()
84.         .required("Number is required")
85.         .min(0, "Number must be at least 0")
86.         .max(100, "Number must be at most 100"),
87. });
88.
89. const handleSubmit = (values, { setSubmitting }) => {
90.     alert("Form submitted successfully!");
91.     setSubmitting(false);
92. };
93.
94. return (
95.     <div className="form-window">
96.         <div className="form-container">
97.             <div className="form-title">Example form</div>
98.             <Formik
99.                 initialValues={initialValues}
100.                 validationSchema={validationSchema}

```

```

101.         onSubmit={handleSubmit}
102.     >
103.         {{{ isSubmitting }}} => (
104.             <Form>
105.                 { /* Name */}
106.                 <label className="form-label"
107.                     htmlFor="name">
108.                     Name
109.                 </label>
110.                 <Field
111.                     className="form-field"
112.                     type="text"
113.                     id="name"
114.                     name="name"
115.                     placeholder="James"
116.                 />
117.                 <ErrorMessage
118.                     className="form-error-message"
119.                     name="name"
120.                     component="div"
121.                 />
122.                 { /* Email */}
123.                 <label className="form-label"
124.                     htmlFor="email">
125.                     Email
126.                 </label>
127.                 <Field
128.                     className="form-field"
129.                     type="text"
130.                     id="email"
131.                     name="email"
132.                     placeholder="example@mail.com"
133.                 />
134.                 <ErrorMessage
135.                     className="form-error-message"
136.                     name="email"
137.                     component="div"
138.                 />
139.                 { /* Phone */}
140.                 <label className="form-label"
141.                     htmlFor="phone">
142.                     Phone number
143.                 </label>
144.                 <Field
145.                     className="form-field"
146.                     type="text"
147.                     id="phone"
148.                     name="phone"
149.                     placeholder="+380yyxxxxxxx"
150.                 />
151.                 <ErrorMessage
152.                     className="form-error-message"
153.                     name="phone"
154.                     component="div"
155.                 />

```

```

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>

```

```

258.     wrapper">
259.         <Field
260.             className="form-checkbox"
261.             type="checkbox"
262.             id="hobbySport"
263.             name="hobbySport"
264.         />
265.         <label
266.             className="form-selection-
label"
267.             htmlFor="hobbySport"
268.         >
269.             Sport
270.         </label>
271.     </div>
272.
273.     {/* Birthday */}
274.     <label className="form-label"
htmlFor="birthday">
275.         Birthday
276.     </label>
277.     <Field
278.         className="form-field"
279.         type="date"
280.         id="birthday"
281.         name="birthday"
282.     />
283.     <ErrorMessage
284.         className="form-error-message"
285.         name="birthday"
286.         component="div"
287.     />
288.
289.     {/* Time */}
290.     <label className="form-label"
htmlFor="time">
291.         When can we call you?
292.     </label>
293.     <Field
294.         className="form-field"
295.         type="time"
296.         id="time"
297.         name="time"
298.     />
299.     <ErrorMessage
300.         className="form-error-message"
301.         name="time"
302.         component="div"
303.     />
304.
305.     {/* Number */}
306.     <label className="form-label"
htmlFor="number">
307.         Pick a number from 0 to 100
308.     </label>
309.     <Field
310.         className="form-field"

```

```

311.         type="number"
312.         id="number"
313.         name="number"
314.         min="0"
315.         max="100"
316.     />
317. <ErrorMessage
318.     className="form-error-message"
319.     name="number"
320.     component="div"
321. />
322.
323.     {/* Color */}
324. <label className="form-label"
325.     htmlFor="color">
326.         Your favorite color
327. </label>
328. <Field
329.     className="form-field"
330.     type="color"
331.     id="color"
332.     name="color"
333. />
334.
335.     {/* Rating */}
336. <label className="form-label"
337.     htmlFor="rating">
338.         Rate this form on a scale from 0
339.         to 10
340. </label>
341. <Field
342.     className="form-field"
343.     type="range"
344.     id="rating"
345.     name="rating"
346.     min="0"
347.     max="10"
348. />
349.
350.     {/* Season */}
351. <label className="form-label"
352.     htmlFor="season">
353.         Your favorite season
354. </label>
355. <Field
356.     className="form-field"
357.     as="select"
358.     id="season"
359.     name="season"
360. >
361.     <option
362.         value="summer">Summer</option>
363.     <option
364.         value="autumn">Autumn</option>
365.     <option
366.         value="winter">Winter</option>
367.     <option
368.         value="spring">Spring</option>

```

```

361.                                     </Field>
362.
363.                                     { /* Comment */ }
364.                                     <label className="form-label"
      htmlFor="comment">
365.                                         Comment
366.                                     </label>
367.                                     <Field
368.                                         className="form-textarea"
369.                                         as="textarea"
370.                                         id="comment"
371.                                         name="comment"
372.                                         placeholder="Leave feedback
      about the form"
373.                                     />
374.
375.                                     { /* Submit */ }
376.                                     <button
377.                                         className="form-button"
378.                                         type="submit"
379.                                         disabled={isSubmitting}
380.                                     >
381.                                         Submit
382.                                     </button>
383.                                     </Form>
384.                                     )}
385.                                     </Formik>
386.                                 </div>
387.                             </div>
388.                         );
389.                 }
390.
391. 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",

```

```

47.         "You must be at least 13 years old and not older
48.         than 100 years (because it is impossible)",
49.         (value) => {
50.             if (!value) return false;
51.             const today = new Date();
52.             const birthDate = new Date(value);
53.             const age = today.getFullYear() -
54.             birthDate.getFullYear();
55.             const monthDiff = today.getMonth() -
56.             birthDate.getMonth();
57.             if (
58.                 monthDiff < 0 ||
59.                 (monthDiff === 0 &&
60.                 today.getDate() <
61.                 birthDate.getDate())
62.             ) {
63.                 return age - 1 >= 13 && age - 1 <= 100;
64.             }
65.             return age >= 13 && age <= 100;
66.         },
67.         time: yup.string().required("Time is required"),
68.         number: yup
69.             .number()
70.             .required("Number is required")
71.             .min(0, "Number must be at least 0")
72.             .max(100, "Number must be at most 100"),
73.     });
74.
75.     const {
76.         register,
77.         handleSubmit,
78.         control,
79.         formState: { errors },
80.     } = useForm({
81.         resolver: yupResolver(validationSchema),
82.         defaultValues: {
83.             name: "",
84.             email: "",
85.             phone: "",
86.             password: "",
87.             gender: "",
88.             hobbyReading: false,
89.             hobbyDrawing: false,
90.             hobbySport: false,
91.             birthday: null,
92.             time: null,
93.             number: 0,
94.             color: "#EDBB36",
95.             rating: 10,
96.             season: "summer",
97.             comment: "",
98.         },
99.     });
100.
101.     const onSubmit = (data) => {
102.         alert("Form submitted successfully!");
103.         console.log(data);

```

```

101.     };
102.
103.     return (
104.         <div className="form-window">
105.             <div className="form-container">
106.                 <div className="form-title">Example form</div>
107.
108.                 <form onSubmit={handleSubmit(onSubmit)}>
109.                     { /* Name */
110.                     <label className="form-label"
111.                         htmlFor="name">
112.                         Name
113.                     </label>
114.                     <input
115.                         className="form-field"
116.                         type="text"
117.                         id="name"
118.                         {...register("name")}
119.                         placeholder="James"
120.                     />
121.                     {errors.name && (
122.                         <div className="form-error-message">
123.                             {errors.name.message}
124.                         </div>
125.                     )}
126.                     { /* Email */
127.                     <label className="form-label"
128.                         htmlFor="email">
129.                         Email
130.                     </label>
131.                     <input
132.                         className="form-field"
133.                         type="text"
134.                         id="email"
135.                         {...register("email")}
136.                         placeholder="example@mail.com"
137.                     />
138.                     {errors.email && (
139.                         <div className="form-error-message">
140.                             {errors.email.message}
141.                         </div>
142.                     )}
143.                     { /* Phone */
144.                     <label className="form-label"
145.                         htmlFor="phone">
146.                         Phone number
147.                     </label>
148.                     <input
149.                         className="form-field"
150.                         type="text"
151.                         id="phone"
152.                         {...register("phone")}
153.                         placeholder="+380yyxxxxxxx"
154.                     />
155.                     {errors.phone && (

```

```

156.         {errors.phone.message}
157.     </div>
158. }}
159.
160.     {/ * Password */}
161.     <label className="form-label"
        htmlFor="password">
162.         Password
163.     </label>
164.     <input
165.         className="form-field"
166.         type="password"
167.         id="password"
168.         {...register("password")}
169.         placeholder="At least 8 characters"
170.     />
171.     {errors.password && (
172.         <div className="form-error-message">
173.             {errors.password.message}
174.         </div>
175.     )}
176.
177.     {/ * Gender */}
178.     <div className="form-label">Gender</div>
179.     <div className="form-selection-wrapper">
180.         <input
181.             className="form-radio"
182.             type="radio"
183.             id="male"
184.             value="male"
185.             {...register("gender")}
186.         />
187.         <label className="form-selection-label"
        htmlFor="male">
188.             Male
189.         </label>
190.     </div>
191.     <div className="form-selection-wrapper">
192.         <input
193.             className="form-radio"
194.             type="radio"
195.             id="female"
196.             value="female"
197.             {...register("gender")}
198.         />
199.         <label
200.             className="form-selection-label"
201.             htmlFor="female"
202.         >
203.             Female
204.         </label>
205.     </div>
206.     <div className="form-selection-wrapper">
207.         <input
208.             className="form-radio"
209.             type="radio"
210.             id="other"
211.             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

```

```

322. </label>
323. <Controller
324.     name="color"
325.     control={control}
326.     render={({ field }) => (
327.         <input
328.             className="form-field"
329.             type="color"
330.             id="color"
331.             {...field}
332.         />
333.     )}
334. />
335.
336. {/* Rating */}
337. <label className="form-label"
338.     htmlFor="rating">
339.     Rate this form on a scale from 0 to 10
340. </label>
341. <input
342.     className="form-field"
343.     type="range"
344.     id="rating"
345.     {...register("rating")}
346.     min="0"
347.     max="10"
348. />
349.
350. {/* Season */}
351. <label className="form-label"
352.     htmlFor="season">
353.     Your favorite season
354. </label>
355. <select
356.     className="form-field"
357.     id="season"
358.     {...register("season")}
359. >
360.     <option value="summer">Summer</option>
361.     <option value="autumn">Autumn</option>
362.     <option value="winter">Winter</option>
363.     <option value="spring">Spring</option>
364. </select>
365.
366. {/* Comment */}
367. <label className="form-label"
368.     htmlFor="comment">
369.     Comment
370. </label>
371. <textarea
372.     className="form-textarea"
373.     id="comment"
374.     {...register("comment")}
375.     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",

```

```

46.         "You must be at least 13 years old and not older
         than 100 years (because it is impossible)",
47.         (value) => {
48.             if (!value) return false;
49.             const today = new Date();
50.             const birthDate = new Date(value);
51.             const age = today.getFullYear() -
         birthDate.getFullYear();
52.             const monthDiff = today.getMonth() -
         birthDate.getMonth();
53.             if (
54.                 monthDiff < 0 ||
55.                 (monthDiff === 0 &&
56.                     today.getDate() <
         birthDate.getDate())
57.             ) {
58.                 return age - 1 >= 13 && age - 1 <= 100;
59.             }
60.             return age >= 13 && age <= 100;
61.         }
62.     ),
63.     time: yup.string().required("Time is required"),
64.     number: yup
65.         .number()
66.         .required("Number is required")
67.         .min(0, "Number must be at least 0")
68.         .max(100, "Number must be at most 100"),
69. });
70.
71.     const validate = async (values) => {
72.         try {
73.             await validationSchema.validate(values, {
         abortEarly: false });
74.         } catch (err) {
75.             if (err instanceof yup.ValidationError) {
76.                 return err.inner.reduce((errors, error) => {
77.                     errors[error.path] = error.message;
78.                     return errors;
79.                 }, {});
80.             }
81.         }
82.     };
83.
84.     const onSubmit = async (values, form) => {
85.         alert("Form submitted successfully!");
86.         form.reset();
87.     };
88.
89.     return (
90.         <div className="form-window">
91.             <div className="form-container">
92.                 <div className="form-title">Example form</div>
93.                 <Form
94.                     onSubmit={onSubmit}
95.                     validate={validate}
96.                     initialValues={{
97.                         name: "",
98.                         email: "",

```

```

99.         phone: "",
100.        password: "",
101.        gender: "",
102.        hobbyReading: false,
103.        hobbyDrawing: false,
104.        hobbySport: false,
105.        birthday: null,
106.        time: null,
107.        number: 0,
108.        color: "#EDBB36",
109.        rating: 10,
110.        season: "summer",
111.        comment: "",
112.    }}
113.    render={({ handleSubmit, submitError }) => (
114.        <form onSubmit={handleSubmit}>
115.            {submitError && (
116.                <div className="form-error-
117.                    message">
118.                        {submitError}
119.                    </div>
120.                )}
121.            {/* Name */}
122.            <Field name="name">
123.                {{{ input, meta }} => (
124.                    <>
125.                        <label
126.                            className="form-
127.                                label"
128.                            htmlFor="name"
129.                        >
130.                            Name
131.                        </label>
132.                        <input
133.                            {...input}
134.                            className="form-
135.                                field"
136.                            id="name"
137.                            placeholder="James"
138.                        />
139.                        {meta.touched &&
140.                            <span
141.                                className="form-error-message">
142.                                    {meta.error}
143.                                </span>
144.                            )}
145.                    </>
146.                )}
147.            </Field>
148.            {/* Email */}
149.            <Field name="email">
150.                {{{ input, meta }} => (

```

```

151.                                     >
152.                                     Email
153.                                     </label>
154.                                     <input
155.                                       {...input}
156.                                       className="form-
      field"
157.                                       id="email"
158.                                       placeholder="example@mail.com"
159.                                     />
160.                                     {meta.touched &&
      meta.error && (
161.                                       <span
      className="form-error-message">
162.                                           {meta.error}
163.                                       </span>
164.                                       )}
165.                                     </>
166.                                   )}
167. </Field>
168. {/* Phone */}
169. <Field name="phone">
170.   {{{ input, meta }} => (
171.     <>
172.       <label
173.         className="form-
      label"
174.         htmlFor="phone"
175.       >
176.         Phone number
177.       </label>
178.       <input
179.         {...input}
180.         className="form-
      field"
181.         id="phone"
182.         placeholder="+380yyxxxxxxx"
183.       />
184.       {meta.touched &&
      meta.error && (
185.         <span
      className="form-error-message">
186.             {meta.error}
187.         </span>
188.         )}
189.       </>
190.     )}
191. </Field>
192. {/* Password */}
193. <Field name="password">
194.   {{{ input, meta }} => (
195.     <>
196.       <label
197.         className="form-
      label"
198.         htmlFor="password"

```



```

245.                                     className="form-
      selection-label"
246.                                     htmlFor="female"
247.                                     >
248.                                     Female
249.                                     </label>
250.                                   </div>
251.                                   )}
252. </Field>
253. <Field name="gender" type="radio"
      value="other">
254.                                   {{{ input }} => (
255.                                   <div className="form-
      selection-wrapper">
256.                                   <input
257.                                   {...input}
258.                                   className="form-
      radio"
259.                                   id="other"
260.                                   />
261.                                   <label
262.                                   className="form-
      selection-label"
263.                                   htmlFor="other"
264.                                   >
265.                                   Other
266.                                   </label>
267.                                   </div>
268.                                   )}
269. </Field>
270. {/* Gender error display */}
271. <Field name="gender">
272.   {{{ meta }} =>
273.     meta.touched && meta.error ?
274.   (
275.     <span className="form-
      error-message">
276.       {meta.error}
277.     </span>
278.   ) : null
279. }
280. </Field>
281. {/* Hobbies */}
282. <div className="form-label">
283.   Choose your hobbies
284. </div>
285. <Field name="hobbyReading"
      type="checkbox">
286.   {{{ input }} => (
287.     <div className="form-
      selection-wrapper">
288.     <input
289.     {...input}
290.     className="form-
      checkbox"
291.     id="hobbyReading"
292.     />
293.     <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

```

```

340.                                     className="form-
      label"
341.                                     htmlFor="birthday"
342.                                     >
343.                                     Birthday
344.                                   </label>
345.                                   <input
346.                                     {...input}
347.                                     className="form-
      field"
348.                                     id="birthday"
349.                                     type="date"
350.                                   />
351.                                   {meta.touched &&
      meta.error && (
352.                                     <span
353.                                       {meta.error}
354.                                     </span>
355.                                   )}
356.                                 </>
357.                               )}
358.                             </Field>
359.                             {/* Time */}
360.                             <Field name="time">
361.                               {{{ input, meta }} => (
362.                                 <>
363.                                   <label
364.                                     className="form-
      label"
365.                                     htmlFor="time"
366.                                   >
367.                                     When can we call
      you?
368.                                   </label>
369.                                   <input
370.                                     {...input}
371.                                     className="form-
      field"
372.                                     id="time"
373.                                     type="time"
374.                                   />
375.                                   {meta.touched &&
      meta.error && (
376.                                     <span
377.                                       {meta.error}
378.                                     </span>
379.                                   )}
380.                                 </>
381.                               )}
382.                             </Field>
383.                             {/* Number */}
384.                             <Field name="number">
385.                               {{{ input, meta }} => (
386.                                 <>
387.                                   <label

```

```

388.                                     className="form-
      label"
389.                                     htmlFor="number"
390.                                     >
391.                                     Pick a number from 0
      to 100
392.                                     </label>
393.                                     <input
394.                                       {...input}
395.                                       className="form-
      field"
396.                                       id="number"
397.                                       type="number"
398.                                       min="0"
399.                                       max="100"
400.                                     />
401.                                     {meta.touched &&
      meta.error && (
402.                                     <span
      className="form-error-message">
403.                                       {meta.error}
404.                                     </span>
405.                                     )}
406.                                     </>
407.                                     )}
408.                                     </Field>
409.                                     {/* Color */}
410.                                     <Field name="color">
411.                                       {{{ input }} => (
412.                                         <>
413.                                           <label
414.                                             className="form-
      label"
415.                                             htmlFor="color"
416.                                           >
417.                                             Your favorite color
418.                                           </label>
419.                                           <input
420.                                             {...input}
421.                                             className="form-
      field"
422.                                             id="color"
423.                                             type="color"
424.                                           />
425.                                         </>
426.                                       )}
427.                                     </Field>
428.                                     {/* Rating */}
429.                                     <Field name="rating">
430.                                       {{{ input }} => (
431.                                         <>
432.                                           <label
433.                                             className="form-
      label"
434.                                             htmlFor="rating"
435.                                           >
436.                                             Rate this form on a
      scale from 0 to

```

```

437.                                     10
438.                                   </label>
439.                                   <input
440.                                     {...input}
441.                                     className="form-
      field"
442.                                     id="rating"
443.                                     type="range"
444.                                     min="0"
445.                                     max="10"
446.                                   />
447.                                 </>
448.                               )}
449. </Field>
450. {/* Season */}
451. <Field name="season">
452.   {{{ input }} => (
453.     <>
454.       <label
455.         className="form-
      label"
456.         htmlFor="season"
457.       >
458.         Your favorite season
459.       </label>
460.       <select
461.         {...input}
462.         className="form-
      field"
463.         id="season"
464.       >
465.         <option
466.           value="summer">
467.             Summer
468.           </option>
469.           <option
470.             value="autumn">
471.               Autumn
472.             </option>
473.             <option
474.               value="winter">
475.                 Winter
476.               </option>
477.               <option
478.                 value="spring">
479.                   Spring
480.                 </option>
481.               </select>
482.             </>
483.           )}
484.         </Field>
485.         {/* Comment */}
486.         <Field name="comment">
487.           {{{ input }} => (
488.             <>
489.               <label
490.                 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();

```

```

45.             if (
46.                 monthDiff < 0 ||
47.                 (monthDiff === 0 && today.getDate() <
    birthDate.getDate())
48.             ) {
49.                 return age - 1 >= 13 && age - 1 <= 100;
50.             }
51.             return age >= 13 && age <= 100;
52.         }
53.     ),
54.     time: yup.string().required("Time is required"),
55.     number: yup
56.         .number()
57.         .required("Number is required")
58.         .min(0, "Number must be at least 0")
59.         .max(100, "Number must be at most 100"),
60. });
61.
62. const validate = (values) => {
63.     const errors = {};
64.     try {
65.         validationSchema.validateSync(values, { abortEarly:
false });
66.     } catch (err) {
67.         if (err.inner) {
68.             err.inner.forEach((validationError) => {
69.                 errors[validationError.path] =
validationError.message;
70.             });
71.         }
72.     }
73.     return errors;
74. };
75.
76. const renderField = ({
77.     input,
78.     label,
79.     type,
80.     meta: { touched, error },
81.     placeholder,
82.     min,
83.     max,
84. }) => (
85.     <div>
86.         <label className="form-label">{label}</label>
87.         <input
88.             {...input}
89.             type={type}
90.             className="form-field"
91.             placeholder={placeholder}
92.             min={min}
93.             max={max}
94.         />
95.         {touched && error && (
96.             <span className="form-error-message">{error}</span>
97.         )}
98.     </div>
99. );

```

```

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"
111.                 key={option.value}>
112.                 <input
113.                     {...input}
114.                     type="radio"
115.                     value={option.value}
116.                     checked={input.value === option.value}
117.                     className="form-radio"
118.                 />
119.                 <label className="form-selection-
120. label">{option.label}</label>
121.                 </div>
122.             )})
123.         {touched && error && (
124.             <span className="form-error-message">{error}</span>
125.         )}
126.     </div>
127. );
128.
129. const renderCheckboxGroup = ({ input, options, label }) => (
130.     <div>
131.         <div className="form-label">{label}</div>
132.         {options.map((option) => (
133.             <div className="form-selection-wrapper"
134.                 key={option.name}>
135.                 <input
136.                     type="checkbox"
137.                     name={option.name}
138.                     checked={input.value.includes(option.name)}
139.                     onChange={(event) => {
140.                         const newValue = [...input.value];
141.                         if (event.target.checked) {
142.                             newValue.push(option.name);
143.                         } else {
144.                             newValue.splice(newValue.indexOf(option.name), 1);
145.                         }
146.                         input.onChange(newValue);
147.                     }}
148.                     className="form-checkbox"
149.                 />
150.                 <label className="form-selection-
151. label">{option.label}</label>
152.                 </div>
153.             )})
154.         </div>
155.     );

```

```

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)}>

```

```

209. <div className="form-container">
210.   <div className="form-title">Example form</div>
211.
212.   <Field
213.     name="name"
214.     type="text"
215.     component={renderField}
216.     label="Name"
217.     placeholder="James"
218.   />
219.   <Field
220.     name="email"
221.     type="text"
222.     component={renderField}
223.     label="Email"
224.     placeholder="example@mail.com"
225.   />
226.   <Field
227.     name="phone"
228.     type="text"
229.     component={renderField}
230.     label="Phone number"
231.     placeholder="+380yyxxxxxxxx"
232.   />
233.   <Field
234.     name="password"
235.     type="password"
236.     component={renderField}
237.     label="Password"
238.     placeholder="At least 8 characters"
239.   />
240.
241.   <Field
242.     name="gender"
243.     component={renderRadioGroup}
244.     label="Gender"
245.     options={[
246.       { value: "male", label: "Male" },
247.       { value: "female", label: "Female" },
248.       { value: "other", label: "Other" },
249.     ]}
250.   />
251.
252.   <Field
253.     name="hobbies"
254.     component={renderCheckboxGroup}
255.     label="Choose your hobbies"
256.     options={[
257.       { name: "hobbyReading", label: "Reading"
258.       },
259.       { name: "hobbyDrawing", label: "Drawing"
260.       },
261.       { name: "hobbySport", label: "Sport" },
262.     ]}
263.   />
264.   <Field
      name="birthday"

```

```

265.         type="date"
266.         component={renderField}
267.         label="Birthday"
268.     />
269. <Field
270.     name="time"
271.     type="time"
272.     component={renderField}
273.     label="When can we call you?"
274. />
275. <Field
276.     name="number"
277.     type="number"
278.     component={renderField}
279.     label="Pick a number from 0 to 100"
280.     min={0}
281.     max={100}
282. />
283. <Field
284.     name="color"
285.     type="color"
286.     component={renderField}
287.     label="Your favorite color"
288. />
289. <Field
290.     name="rating"
291.     type="range"
292.     component={renderField}
293.     label="Rate this form on a scale from 0 to
    10"
294.     min={0}
295.     max={10}
296. />
297.
298. <Field
299.     name="season"
300.     component={renderSelect}
301.     label="Your favorite season"
302.     options={[
303.         { value: "summer", label: "Summer" },
304.         { value: "autumn", label: "Autumn" },
305.         { value: "winter", label: "Winter" },
306.         { value: "spring", label: "Spring" },
307.     ]}
308. />
309.
310. <Field
311.     name="comment"
312.     component={renderTextarea}
313.     label="Comment"
314.     placeholder="Leave feedback about the form"
315. />
316.
317. <button className="form-button" type="submit">
318.     Submit
319. </button>
320. </div>
321. </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

Example form

Name

Email

Phone number

Password

Gender

Male Female Other

Choose your hobbies

Reading Drawing Sport

Birthday



When can we call you?



Pick a number from 0 to 100

Your favorite color



Rate this form on a scale from 0 to 10



Your favorite season



Comment