Специфікація вимог до програмного забезпечення

для

Експериментального програмного комплексу для дослідження ефективності підходів до розробки форм у React

Затверджено версією 1.0

Підготовлено Дудником Олександром Олеговичем

Харківський національний університет радіоелектроніки

12.06.2025

Зміст

Зміст ii

Журнал версій (Revision History) ii

1. Введення 1

1.1 Мета 1

1.2 Конвенції про документ 1

1.3 Цільова аудиторія і пропозиції з читання 1

1.4 Область проекту 2

1.5 Посилання 2

2. Загальний опис 2

2.1 Перспектива продукту 2

2.2 Особливості продукту 3

2.3 Класи та характеристики користувачів 3

2.4 Операційне середовище 3

2.5 Обмеження щодо проектування та впровадження 4

2.6 Документація користувача 4

2.7 Припущення та залежності 4

3. Вимоги до зовнішнього інтерфейсу 5

3.1 Інтерфейси користувача 5

3.2 Апаратні інтерфейси 5

3.3 Інтерфейси програмного забезпечення 5

3.4 Інтерфейси зв’язку 6

4. Функції системи 6

4.1 Заповнення та відправка форми 6

5. Нефункціональні вимоги 6

5.1 Вимоги до продуктивності 6

5.2 Вимоги щодо неушкодженості (техніки безпеки) 7

5.3 Вимоги до безпеки 7

5.4 Атрибути якості програмного забезпечення 7

5.5 Бізнес правила 7

6. Інші вимоги 7

Журнал версій (Revision History)

|  |  |  |  |
| --- | --- | --- | --- |
| **Ім’я** | **Дата** | **Причини для змін** | **Версія** |
| Дудник Олександр Олегович | 12.06.2025 | Перша редакція | 1.0 |
|  |  |  |  |

# Введення

## Мета

Метою цього документа є специфікація вимог до експериментального програмного комплексу, розробленого для дослідження ефективності різних підходів до розробки форм у React.

Система складається з п'яти ідентичних за функціоналом додатків, кожен з яких реалізований з використанням різного підходу: чистий React, Formik, React Hook Form, Final Form та Redux Form.

Документ призначений для формального опису функціональних та нефункціональних вимог до системи, забезпечення основи для розробки та тестування застосунків в контексті дипломного дослідження.

## Конвенції про документ

Документ оформлено відповідно до стандарту IEEE SRS (Software Requirements Specification).

Документ написаний українською мовою з використанням англійських термінів для назв технологій та бібліотек. Спеціальні терміни та абревіатури пояснюються при першому використанні. Ключові терміни, поняття та категорії, на які слід звернути увагу, виділяються жирним шрифтом. Пріоритети вимог визначаються як: Високий, Середній, Низький.

## Цільова аудиторія і пропозиції з читання

Цей документ призначений для наступних категорій читачів:

* **Науковий керівник та рецензенти**, які здійснюють оцінювання технічного рівня, повноти реалізації та відповідності проєкту поставленим завданням.
* **Розробники**, зацікавлені в аналізі ефективності різних підходів до реалізації форм у React, а також у використанні даних експерименту для подальших досліджень або вибору найбільш ефективного рішення для власних проектів.
* **Студенти та дослідники**, що вивчають сучасні підходи до побудови інтерфейсів у React і прагнуть зрозуміти практичні аспекти використання різних бібліотек для форм.
* **Тестувальники ПЗ**, яким це буде корисно для розуміння специфікацій реалізованих програм, умов тестування та очікуваної поведінки систем.

Для ефективного використання цього документа рекомендується ознайомлюватися з ним у наступному порядку:

* Розділ 1 - для ознайомлення з метою документа, областю проекту та загальними конвенціями.
* Розділ 2 - для розуміння загальної архітектури системи, її основних особливостей та операційного середовища.
* Розділ 3 - для вивчення вимог до користувацьких та програмних інтерфейсів системи.
* Розділ 4 - для детального ознайомлення з функціональними можливостями та вимогами.
* Розділ 5 - для розуміння нефункціональних вимог, включаючи продуктивність, безпеку та атрибути якості.

## Область проекту

Дослідницька система призначена для практичного порівняння п'яти різних підходів до розробки форм у React-додатках шляхом створення функціонально ідентичних демонстраційних програм.

Основні цілі системи:

* Надати практичну демонстрацію реальних сценаріїв використання кожного з п'яти підходів до розробки форм
* Забезпечити об'єктивну основу та надати можливість для практичного порівняння ефективності різних технологічних рішень
* Створення бази для подальших досліджень у сфері веб-розробки

Функціональні межі системи:

Кожен з п'яти додатків реалізує ідентичний набір функцій для роботи з формами, включаючи валідацію даних, обробку помилок, і відправку форм. Система не включає серверну частину та зосереджена виключно на клієнтській реалізації.

Обмеження:

Система призначена виключно для дослідницьких та навчальних цілей і не передбачає використання у production-середовищі.

## Посилання

IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications.

ISO/IEC 25010:2011 - Systems and software Quality Requirements and Evaluation (SQuaRE).

React Documentation - https://react.dev/reference/react

Formik Documentation - https://formik.org/docs

React Hook Form Documentation - https://react-hook-form.com/docs

Final Form Documentation - https://final-form.org/docs/final-form/getting-started

Redux Form Documentation - https://redux-form.com/8.3.0/docs/api/

Evaluation of UI Component Libraries in React Development (Salonen S.) - https://trepo.tuni.fi/handle/10024/148025

Which library should I use? (de la Mora F. L., Nadi S.) - https://doi.org/10.1145/3183399.3183418

Comparison of Redux and React Hooks methods in terms of performance (Pronina D., Kyrychenko I.) - https://api.semanticscholar.org/CorpusID:250625540

# Загальний опис

## Перспектива продукту

Даний програмний продукт є частиною дослідницької роботи, спрямованої на порівняння ефективності різних підходів до реалізації форм у React. Продукт не призначений для комерційного використання, а розроблений виключно для академічних і дослідницьких цілей та слугує експериментальним середовищем для вимірювання кількісних показників, таких як: кількість рядків коду, час першого рендерингу, кількість ререндерів, об’єм споживання пам’яті, час відгуку INP (interaction to next paint), розмір бібліотеки у бандлі.

## Особливості продукту

Основні особливості розробленого програмного забезпечення:

* **Функціональна ідентичність усіх реалізацій.**  
  Усі п’ять версій застосунку мають однакову логіку: ідентичний набір з 13 полів вводу різного типу з однаковими правилами валідації, відображенням помилок, обробкою відправки форми та базовою інтерактивністю.
* **Уніфікований користувацький інтерфейс.**  
  Усі реалізації мають однаковий вигляд інтерфейсу, з однаковою структурою та розташуванням елементів, ідентичними стилями, кольорами та типографікою, що унеможливлює вплив дизайну на результати експерименту.
* **Однакове використання допоміжних бібліотек.**У всіх реалізаціях, де це доречно, використовуються одні й ті самі бібліотеки для допоміжних задач, наприклад, Yup для валідації. Це дозволяє забезпечити рівні умови для порівняння основних підходів до обробки форм, без стороннього впливу додаткових факторів.
* **Незалежність додатків.**Кожен додаток функціонує як окрема, самостійна система з окремим робочим середовищем, незалежними системами збірки та розгортання, ізольованими залежностями та конфігураціями.

## Класи та характеристики користувачів

Даний програмний продукт не призначений для використання реальними кінцевими користувачами, і не передбачає інтеграції в продакшн-системи. Він розроблений виключно з дослідницькою метою в межах дипломної роботи.

Оскільки програмний продукт створено в рамках дослідження, основними користувачами є технічні спеціалісти, пов’язані з розробкою веб-застосунків, а також представники академічної спільноти.

**Основні класи користувачів**:

* Студенти та аспіранти - можуть використовувати продукт як базу для власних досліджень у сфері веб-розробки, продуктивності фронтенду та архітектури форм.
* Фронтенд-розробники - зацікавлені у порівнянні підходів до створення форм, а також у виборі оптимального рішення для проєктів з різними вимогами до продуктивності, зручності підтримки та масштабованості.
* Викладачі та науковці - використовують програму як приклад у навчальному процесі або як інструмент для подальших досліджень ефективності бібліотек у React.
* Екзаменаційна комісія та керівник дипломної роботи - вивчають продукт з метою оцінювання результатів роботи, якості реалізації та рівня аналізу, проведеного автором.

Характеристики користувачів:

Базовий рівень технічної обізнаності - середній і вище.

Досвід користування React - від початкового до експертного.

## Операційне середовище

* Операційна система: Windows 10 Pro 22H2
* Браузер: Google Chrome 131.0.6778.265
* Node.js: v16.13.0
* React: 18.2.0
* Yup: 1.6.1

У відповідних реалізаціях:

* Formik: 2.4.6
* React-Hook-Form: 7.56.2
* Final-Form: 4.20.10
* React-Final-Form: 6.5.9
* Redux: 5.0.1
* Redux-Form: 8.3.10

## Обмеження щодо проектування та впровадження

* **Відсутність бекенда** - усі реалізації працюють автономно, без обміну даних з сервером, щоб не впливати на вимірювання продуктивності.
* **Спрощена логіка форми** - форма не містить складних залежностей між полями: немає умовної логіки, динамічного додавання чи видалення полів, вкладених форм тощо.
* **Використання лише базових UI-компонентів** - щоб уникнути впливу стилізації на продуктивність, не використовувалися сторонні UI-бібліотеки.
* **Відсутність адаптації під мобільні пристрої** - інтерфейс не оптимізований для різних розмірів екранів - дослідження зосереджене на функціональній частині та продуктивності.
* **Відсутність локалізації** - усі тексти в інтерфейсі наведені лише англійською мовою - підтримка мультимовності не реалізована.
* **Використання лише Create React App** - усі реалізації створено з однаковим шаблоном CRA без додаткової оптимізації або кастомізації Webpack.

## Документація користувача

Оскільки програмний продукт не орієнтований на кінцевих користувачів і не виконує роль повноцінного веб-застосунку, жодна форма користувацької документації (наприклад, керівництво користувача, інструкції з використання інтерфейсу тощо) не створювалася.

Оскільки продукт має дослідницький характер, передбачено лише технічну документацію, орієнтовану на ознайомлення з реалізацією та метою застосунків.

Наявна документація:

* Цей документ (SRS) – містить повний опис функціоналу, призначення, середовища використання та вимог до програмного забезпечення.
* Пояснювальна записка до кваліфікаційної роботи – містить теоретичні засади, опис архітектури програмного забезпечення, опис програмної реалізації, методику проведення експериментального дослідження, результати дослідження, аналіз отриманих результатів, а також висновки щодо ефективності різних підходів до створення форм у React.

## Припущення та залежності

* Припускається, що застосунки запускатимуться у сучасному веб-браузері, сумісному з останніми версіями React.
* Очікується, що робота відбувається в стабільному середовищі, і під час запуску додатку не буде сторонніх процесів, які значно навантажують систему і можуть вплинути на результати вимірювання.
* Застосунки створені в навчальних та дослідницьких цілях, тож передбачається, що користувач знайомий з базовими поняттями React, розуміє основи компонентного підходу, хуків тощо.
* Всі реалізації припускають, що користувач не намагається зламати форму або вводити некоректні (зловмисні) дані. Захист від XSS або інших типів атак не є предметом дослідження.
* Для запуску застосунків потрібне встановлене середовище Node.js та менеджер пакетів (npm або yarn), оскільки всі форми збираються як окремі React-застосунки.
* Усі форми реалізовані з використанням актуальних стабільних версій відповідних бібліотек на момент проведення дослідження. Результати можуть змінитися при оновленні версій.
* Проєкт не залежить від зовнішніх API або онлайн-сервісів. Усі необхідні залежності зберігаються локально у файловій системі під час розгортання.

# Вимоги до зовнішнього інтерфейсу

## Інтерфейси користувача

Кожен із п’яти варіантів реалізації форми має однаковий інтерфейс користувача.

Інтерфейс включає:

* заголовок форми;
* поля введення для ім’я, адреси електронної пошти, номеру телефону, паролю;
* групу радіо-кнопок;
* чекбокси;
* поле з вибором дати;
* поле з вибором часу;
* числове поле;
* поле для вибору кольору;
* повзунок (від 0 до 10);
* випадаючий список;
* багаторядкове текстове поле;
* кнопку «Submit», що надсилає форму;
* повідомлення про помилки, що з’являються у випадку некоректного введення;
* візуальне підтвердження успішної відправки форми.

Форма розміщена по центру сторінки, компоненти вирівняні вертикально, з рівномірними відступами. Стилізація мінімалістична. Інтерфейс не передбачає складних анімацій чи інтерактивних елементів і оптимізований під використання на десктопі. Всі поля мають відповідні плейсхолдери.

## Апаратні інтерфейси

Система не передбачає взаємодії з будь-якими спеціалізованими апаратними інтерфейсами. Для коректної роботи достатньо стандартного сучасного персонального комп’ютера або ноутбука з підтримкою веб-браузера.

## Інтерфейси програмного забезпечення

Програмне забезпечення реалізоване як вебзастосунок на основі React. Зовнішні програмні інтерфейси (API) не використовуються. Усі функціональні можливості реалізовані виключно на стороні клієнта без звернень до серверу або бази даних. Для реалізації форм використовуються сторонні бібліотеки: Formik, React Hook Form, Final Form, Redux Form.

## Інтерфейси зв’язку

Застосунок не взаємодіє з мережею або сторонніми сервісами під час роботи. Як вебзастосунок, він завантажується через стандартний протокол HTTP/HTTPS із локального вебсервера. Після завантаження вся обробка даних виконується на стороні клієнта без додаткових мережевих запитів.

# Функції системи

## Заповнення та відправка форми

4.1.1 Опис і піоритет

Ця функція дозволяє користувачеві заповнити всі обов’язкові поля форми та надіслати дані через кнопку «Submit». Пріоритет: Високий

4.1.2 Послідовність стимул/відповідь

1. Користувач відкриває сторінку з формою.
2. Вводить значення в усі або деякі поля форми.
3. Натискає кнопку «Submit».
4. Система виконує перевірку валідності введених даних:
   * Якщо всі дані валідні, форма відправляється (локально обробляється), і з’являється повідомлення про успіх.
   * Якщо є помилки, система виводить повідомлення про помилки.

4.1.3 Функціональні вимоги

REQ-1: Форма повинна містити наступні поля: ім’я (текстове поле), email (текстове поле), номер телефону (текстове поле), пароль (поле типу password), стать (радіокнопки), хобі (чекбокси), дата народження (date picker), час зв’язку (time picker), числове поле (0–100), колір (color picker), оцінка форми (повзунок), улюблена пора року (випадаючий список), коментар (textarea).

REQ-2: Усі обов’язкові поля повинні перевірятися на валідність перед надсиланням форми.

REQ-3: У разі помилок введення користувач повинен бачити повідомлення про помилки біля відповідних полів.

REQ-4: Успішне заповнення форми повинно завершуватись повідомленням про успішну відправку.

REQ-5: Усі взаємодії повинні відбуватись без перезавантаження сторінки.

# Нефункціональні вимоги

## Вимоги до продуктивності

Цей застосунок створений виключно як експериментальна платформа для дослідження продуктивності різних підходів до реалізації форм у React. У зв’язку з цим, не встановлюються заздалегідь фіксовані вимоги до продуктивності. Натомість, продуктивність вимірюється емпірично на основі метрик, таких як: час першого рендеру, кількість ререндерів, споживання пам’яті, час відгуку INP, розмір бандлу.

Вимірювання здійснюються з використанням React Profiler, Chrome DevTools та інших інструментів аналізу продуктивності.

Метою дипломної роботи є порівняльний аналіз ефективності реалізацій, а не забезпечення конкретного рівня продуктивності.

## Вимоги щодо неушкодженості (техніки безпеки)

* У зв’язку з відсутністю серверної частини, система не виконує зберігання або передачу даних, отже ризики несанкціонованого доступу мінімальні.
* Проте, форма не повинна допускати JavaScript-ін’єкції або обробку потенційно небезпечного HTML-вводу (наприклад, через захист від oninput="..." або <script> в текстових полях).
* Усі поля введення повинні бути оброблені без виклику критичних помилок або збоїв у роботі інтерфейсу.

## Вимоги до безпеки

* Форма працює виключно локально, тому вимоги до збереження конфіденційності або аутентифікації не застосовуються.
* Всі введені дані залишаються в межах сесії браузера та не передаються третім сторонам.
* Забезпечується базовий захист від XSS-поведінки через правильну обробку/відображення даних (відсутність HTML-ін’єкцій при відображенні введеного тексту).

## Атрибути якості програмного забезпечення

**Простота**: код реалізовано з дотриманням принципів читабельності, використано стандартизовані підходи бібліотек (Formik, React Hook Form тощо).

**Модульність**: форма реалізована як окремий компонент, що спрощує її заміну та повторне використання.

**Тестованість**: кожна форма підтримує юніт-тестування логіки валідації (опціонально).

**Портативність**: застосунок може бути запущений у будь-якому сучасному браузері без додаткових налаштувань.

**Масштабованість**: структура реалізації дозволяє легко розширювати форму новими полями.

## Бізнес правила

Застосунок не містить бізнес-логіки як такої.

Єдина логіка, яку можна умовно віднести до «бізнес правил» - це правила валідації полів (наприклад: мінімальна довжина пароля — 8 символів, номер телефону має бути у форматі +380xxxxxxxxx, поле Email має бути валідною адресою тощо).

Ці правила однакові у всіх реалізаціях і служать основою для порівняння функціоналу форм.

# Інші вимоги

На момент підготовки документації додаткові вимоги до системи відсутні.

Appendix A: Глосарій

SRS (Software Requirements Specification) – документ, що містить повний опис вимог до програмного забезпечення.

API (Application Programming Interface) – програмний інтерфейс для доступу до даних та функцій системи.

INP (Interaction to Next Paint) – метрика продуктивності, що вимірює затримку між взаємодією користувача та відображенням оновленого контенту на екрані.

React Profiler – інструмент для аналізу продуктивності React-застосунків, що дозволяє відстежувати час рендерингу компонентів.

Chrome DevTools – набір інструментів розробника у браузері Google Chrome для аналізу продуктивності, налагодження та оптимізації веб-застосунків.

Formik – бібліотека для роботи з формами у React, що спрощує управління станом форм і валідацію даних.

React Hook Form – бібліотека для управління формами у React, яка використовує хуки для оптимізації продуктивності.

Final Form – бібліотека для керування формами, що підтримує декларативний підхід і оптимізує роботу з формами без зайвих ререндерів.

React-Final-Form – реалізація Final Form для React, що забезпечує ефективне управління станом форм.

Redux Form – бібліотека для управління формами у React з використанням Redux, що дозволяє централізовано зберігати стан форми.

Create React App (CRA) – офіційний шаблон для швидкого створення React-застосунків, що містить базову конфігурацію Webpack.

Yup – бібліотека для валідації схем JavaScript-об'єктів, часто використовується у формах React.

Бандл (bundle size) – загальний розмір усіх файлів JavaScript, що завантажуються під час роботи веб-застосунку.

Ререндер (rerender) – повторне відображення компонентів у React, що може впливати на продуктивність застосунку.

Node.js – серверне середовище для виконання JavaScript поза браузером, використовується для розробки фронтенд-інструментів та збірки React-додатків.

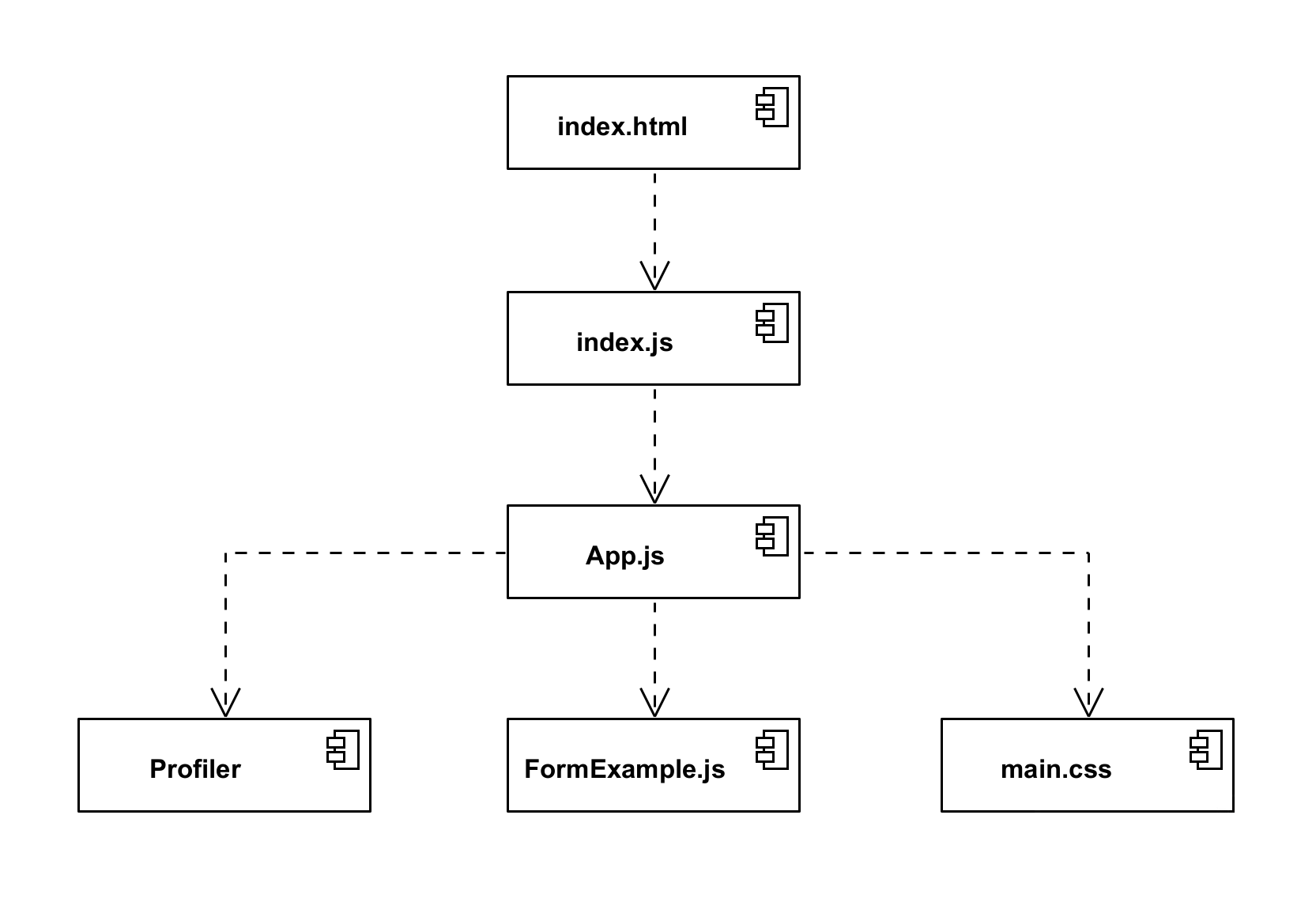
npm (Node Package Manager) – менеджер пакетів для JavaScript, використовується для встановлення залежностей у React-проєктах.

yarn – альтернативний менеджер пакетів для JavaScript, оптимізований для швидкої роботи з залежностями.

XSS (Cross-Site Scripting) – вид атак, що використовує вразливості у веб-застосунках для виконання небезпечного JavaScript-коду.

Appendix B: Моделі аналізу

Діаграма компонентів:



Діаграма активностей:

