**Критерии приема**

Используй этот [шаблон React-проекта](https://github.com/goitacademy/react-homework-template#readme) как стартовую точку своего приложения.

* Созданы репозитории goit-react-hw-02-feedback и goit-react-hw-02-phonebook.
* При сдаче домашней работы есть две ссылки: на исходные файлы и рабочие страницы каждого задания на GitHub Pages.
* При запуске кода задания, в консоли нет ошибок и предупреждений.
* Для каждого компонента есть отдельный файл в папке src/components.
* Все что компонент ожидает в виде пропсов, передается ему при вызове.
* JS-код чистый и понятный, используется Prettier.
* Стилизация выполнена CSS-модулями или Styled Components.

**1 - Виджет отзывов**[**​**](https://textbook.edu.goit.global/lms-react-homework/v1/docs/hw-02/#1---%D0%B2%D0%B8%D0%B4%D0%B6%D0%B5%D1%82-%D0%BE%D1%82%D0%B7%D1%8B%D0%B2%D0%BE%D0%B2)

Как и большинство компаний, кафе Expresso собирает отзывы от своих клиентов. Твоя задача - создать приложение для сбора статистики. Есть всего три варианта обратной связи: хорошо, нейтрально и плохо.

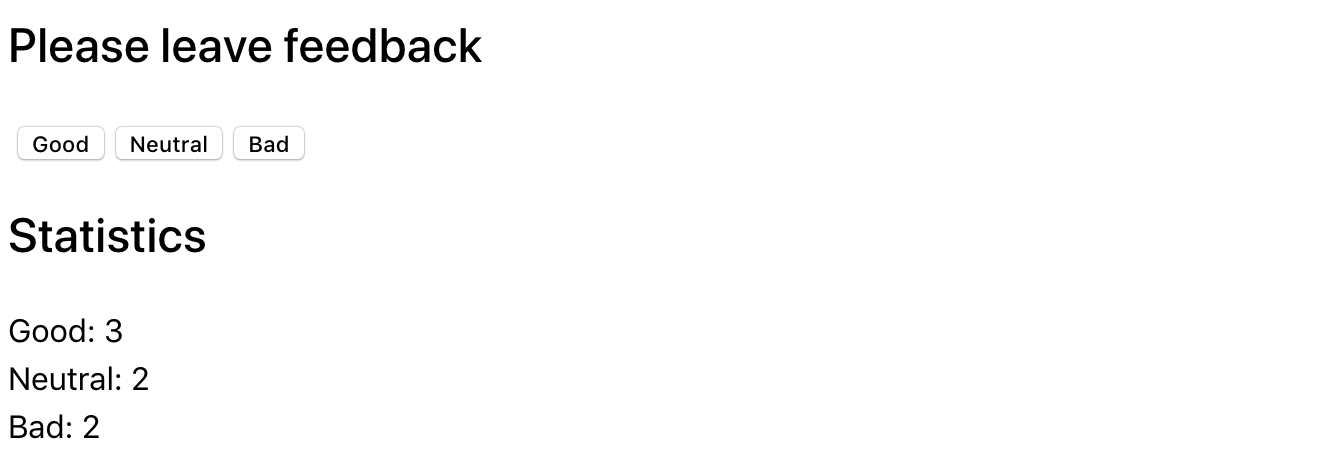
**Шаг 1[​](https://textbook.edu.goit.global/lms-react-homework/v1/docs/hw-02/" \l "%D1%88%D0%B0%D0%B3-1" \o "Прямая ссылка на этот заголовок)**

Приложение должно отображать количество собранных отзывов для каждой категории. Приложение не должно сохранять статистику отзывов между разными сессиями (обновление страницы).

Состояние приложения обязательно должно быть следующего вида, добавлять новые свойства нельзя.

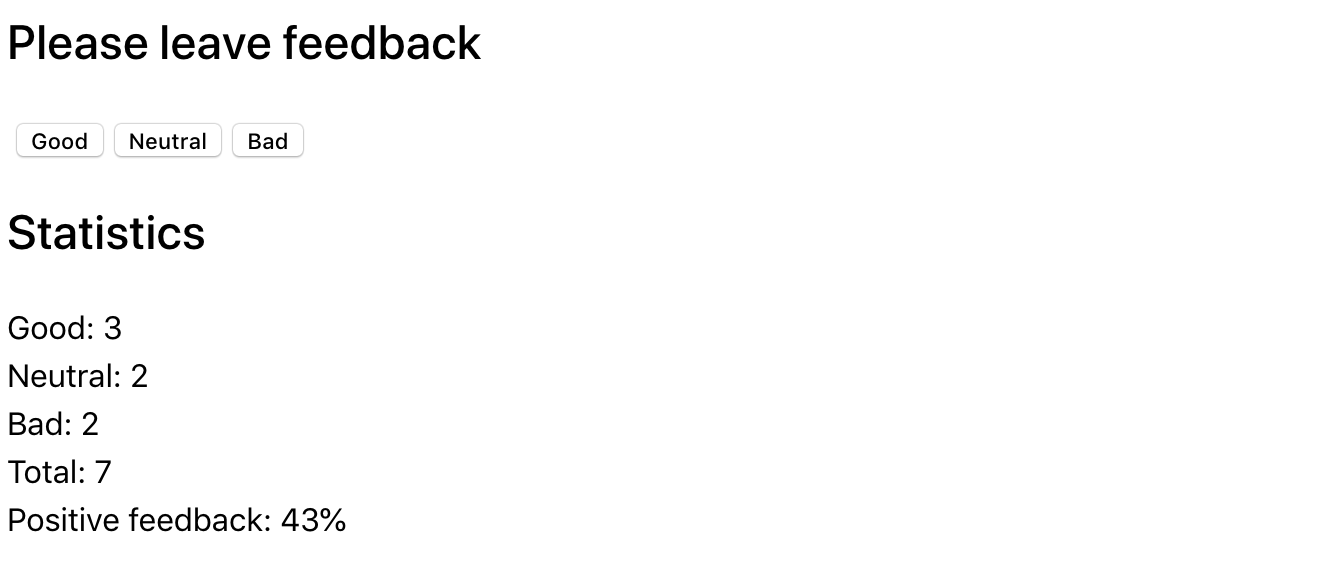
state = {  
 good: 0,  
 neutral: 0,  
 bad: 0  
}

Интерфейс может выглядеть так.



**Шаг 2**[**​**](https://textbook.edu.goit.global/lms-react-homework/v1/docs/hw-02/#%D1%88%D0%B0%D0%B3-2)

Расширь функционал приложения так, чтобы в интерфейсе отображалось больше статистики о собранных отзывах. Добавь отображение общего количества собранных отзывов из всех категорий и процент положительных отзывов. Для этого создай вспомогательные методы countTotalFeedback() и countPositiveFeedbackPercentage(), подсчитывающие эти значения основываясь на данных в состоянии (вычисляемые данные).



**Шаг 3**[**​**](https://textbook.edu.goit.global/lms-react-homework/v1/docs/hw-02/#%D1%88%D0%B0%D0%B3-3)

Выполни рефакторинг приложения. Состояние приложения должно оставаться в корневом компоненте <App>.

* Вынеси отображение статистики в отдельный компонент <Statistics good={} neutral={} bad={} total={} positivePercentage={}>.
* Вынеси блок кнопок в компонент <FeedbackOptions options={} onLeaveFeedback={}>.
* Создай компонент <Section title="">, который рендерит секцию с заголовком и детей (children). Оберни каждый из <Statistics> и <FeedbackOptions> в созданный компонент секции.

**Шаг 4**[**​**](https://textbook.edu.goit.global/lms-react-homework/v1/docs/hw-02/#%D1%88%D0%B0%D0%B3-4)

Расширь функционал приложения так, чтобы блок статистики рендерился только после того, как был собран хотя бы один отзыв. Сообщение об отсутствиии статистики вынеси в компонент <Notification message="There is no feedback">.



**2 - Книга контактов**[**​**](https://textbook.edu.goit.global/lms-react-homework/v1/docs/hw-02/#2---%D0%BA%D0%BD%D0%B8%D0%B3%D0%B0-%D0%BA%D0%BE%D0%BD%D1%82%D0%B0%D0%BA%D1%82%D0%BE%D0%B2)

Напиши приложение хранения контактов телефонной книги.

**Шаг 1**[**​**](https://textbook.edu.goit.global/lms-react-homework/v1/docs/hw-02/#%D1%88%D0%B0%D0%B3-1-1)

Приложение должно состоять из формы и списка контактов. На текущем шаге реализуй добавление имени контакта и отображение списка контактов. Приложение не должно сохранять контакты между разными сессиями (обновление страницы).

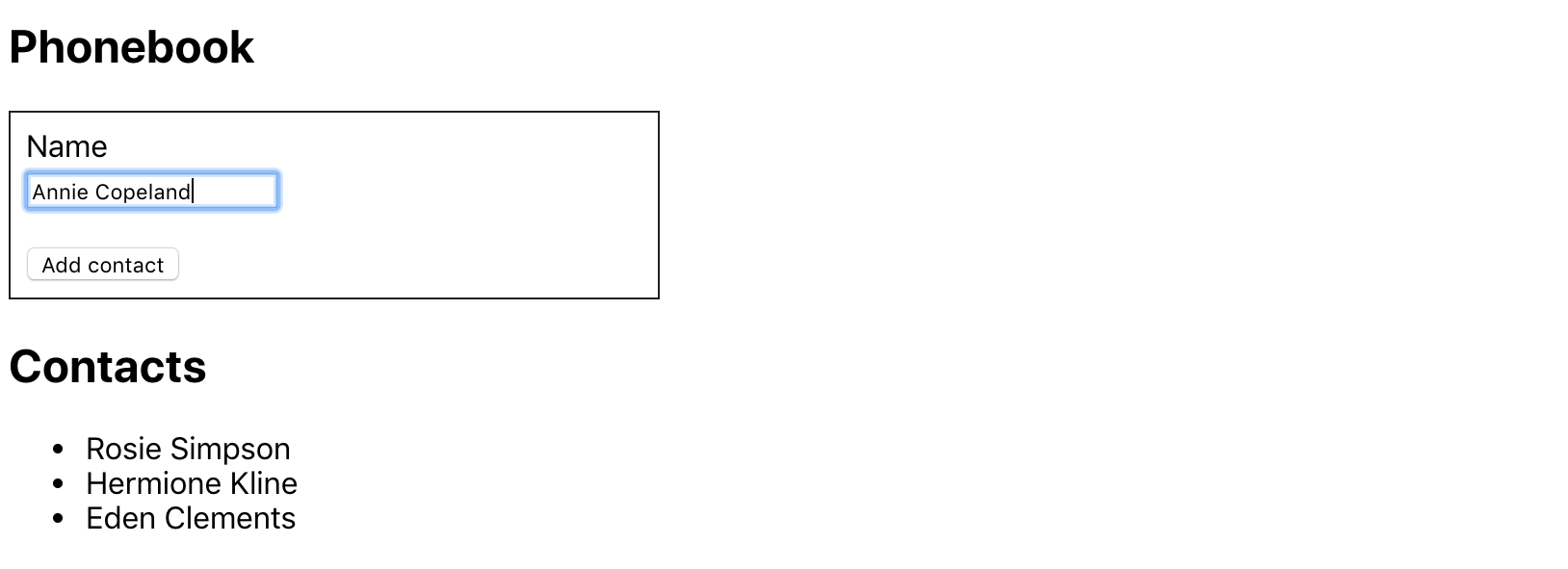
Используй эту разметку инпута с встроенной валидацией для имени контакта.

<input  
 type="text"  
 name="name"  
 pattern="^[a-zA-Zа-яА-Я]+(([' -][a-zA-Zа-яА-Я ])?[a-zA-Zа-яА-Я]\*)\*$"  
 title="Name may contain only letters, apostrophe, dash and spaces. For example Adrian, Jacob Mercer, Charles de Batz de Castelmore d'Artagnan"  
 required  
/>

Состояние хранящееся в родительском компоненте <App> обязательно должно быть следующего вида, добавлять новые свойства нельзя.

state = {  
 contacts: [],  
 name: ''  
}

Каждый контакт должен быть объектом со свойствами name и id. Для генерации идентификаторов используй любой подходящий пакет, например [nanoid](https://www.npmjs.com/package/nanoid" \t "_blank). После завершения этого шага, приложение должно выглядеть примерно так.



**Шаг 2**[**​**](https://textbook.edu.goit.global/lms-react-homework/v1/docs/hw-02/#%D1%88%D0%B0%D0%B3-2-1)

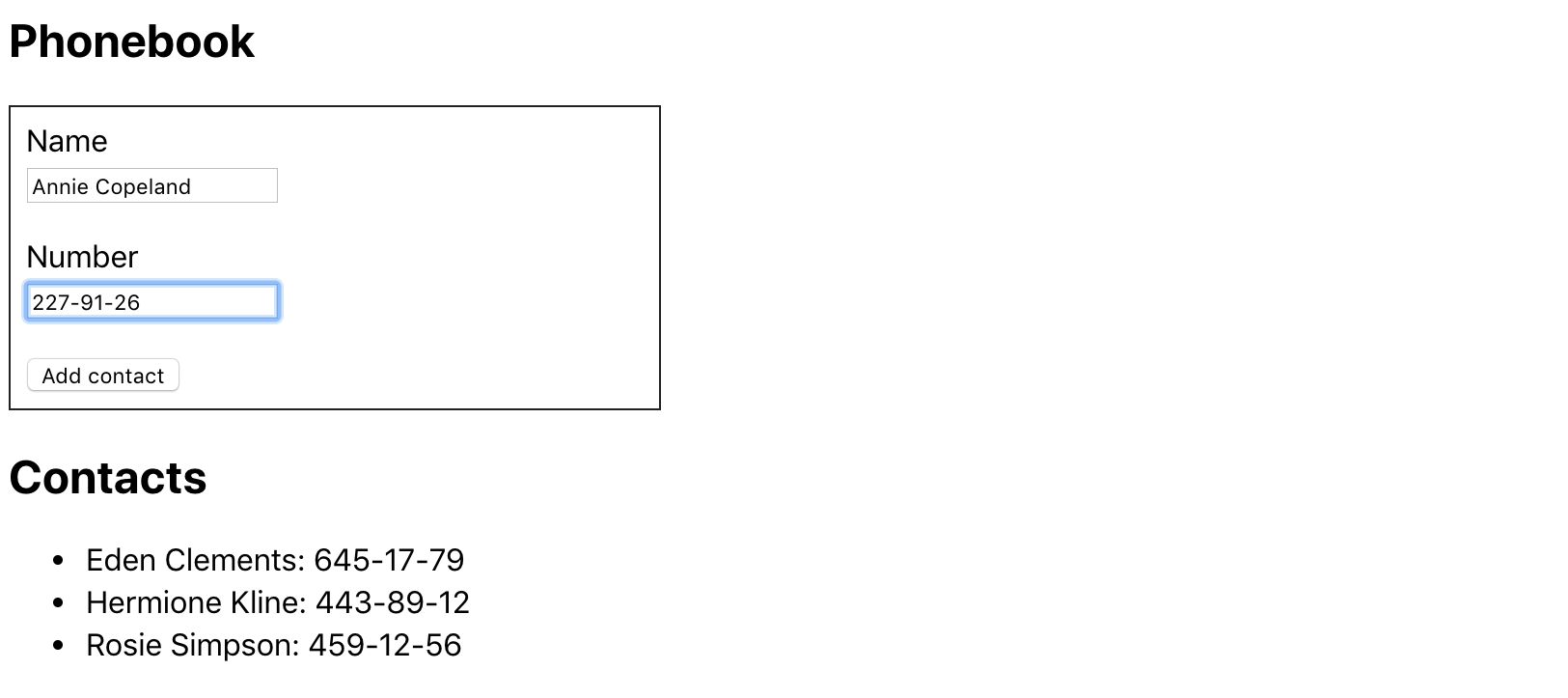
Расширь функционал приложения, позволив пользователям добавлять номера телефонов. Для этого добавь <input type="tel"> в форму, и свойство для хранения его значения в состоянии.

state = {  
 contacts: [],  
 name: '',  
 number: ''  
}

Используй эту разметку инпута с встроенной валидацией для номера контакта.

<input  
 type="tel"  
 name="number"  
 pattern="\+?\d{1,4}?[-.\s]?\(?\d{1,3}?\)?[-.\s]?\d{1,4}[-.\s]?\d{1,4}[-.\s]?\d{1,9}"  
 title="Phone number must be digits and can contain spaces, dashes, parentheses and can start with +"  
 required  
/>

После завершения этого шага, приложение должно выглядеть примерно так.

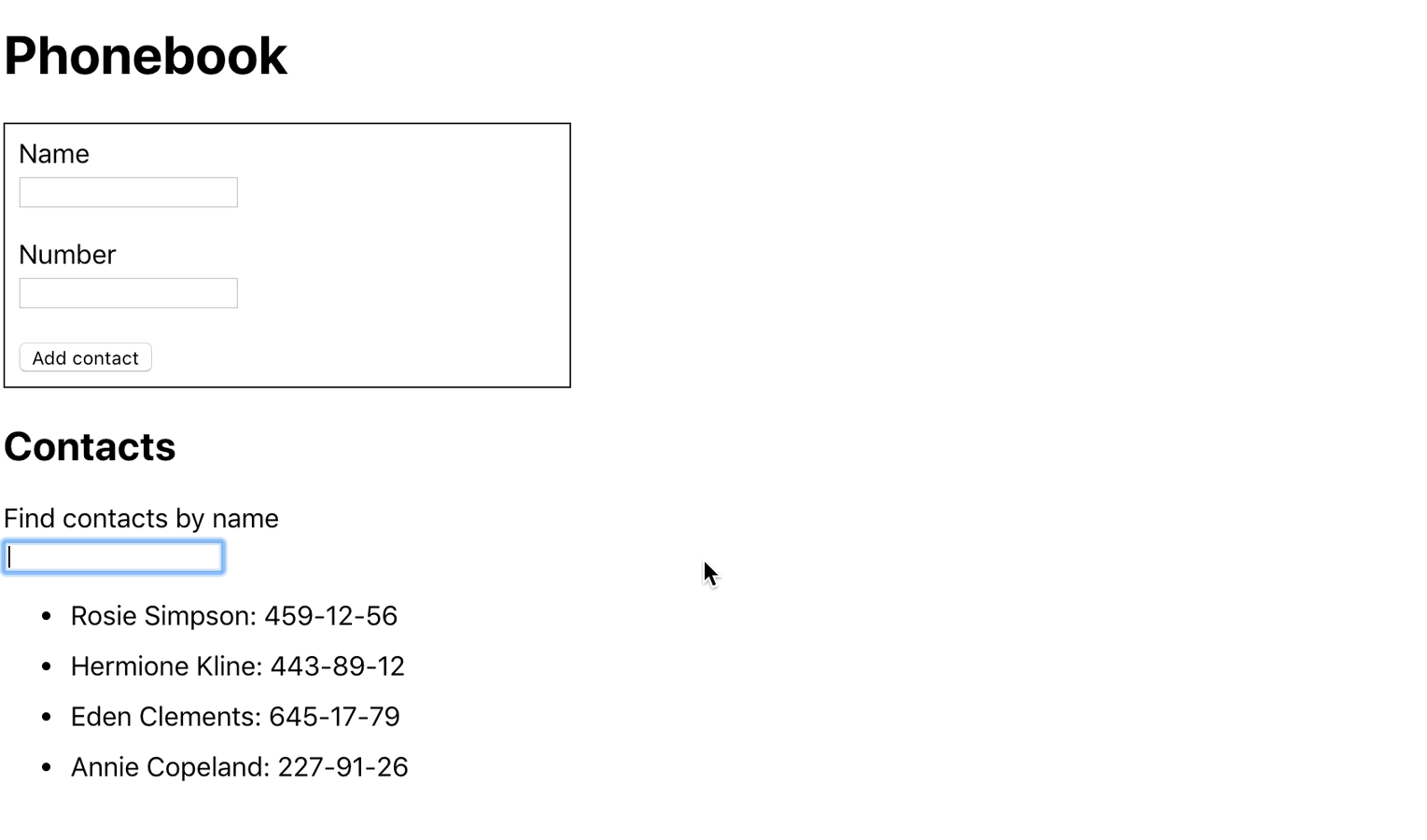


**Шаг 3**[**​**](https://textbook.edu.goit.global/lms-react-homework/v1/docs/hw-02/#%D1%88%D0%B0%D0%B3-3-1)

Добавь поле поиска, которое можно использовать для фильтрации списка контактов по имени.

* Поле поиска это инпут без формы, значение которого записывается в состояние (контролируемый элемент).
* Логика фильтрации должна быть нечувствительна к регистру.

state = {  
 contacts: [],  
 filter: '',  
 name: '',  
 number: ''  
}



Когда мы работаем над новым функционалом, бывает удобно жестко закодировать некоторые данные в состояние. Это избавит от необходимости вручную вводить данные в интерфейсе для тестирования работы нового функционала. Например можно использовать такое начальное состояние.

state = {  
 contacts: [  
 {id: 'id-1', name: 'Rosie Simpson', number: '459-12-56'},  
 {id: 'id-2', name: 'Hermione Kline', number: '443-89-12'},  
 {id: 'id-3', name: 'Eden Clements', number: '645-17-79'},  
 {id: 'id-4', name: 'Annie Copeland', number: '227-91-26'},  
 ],  
 filter: '',  
 name: '',  
 number: ''  
}

**Шаг 4**[**​**](https://textbook.edu.goit.global/lms-react-homework/v1/docs/hw-02/#%D1%88%D0%B0%D0%B3-4-1)

Если твое приложение реализовано в одном компоненте <App>, выполни рефакторинг, выделив подходящие части в отдельные компоненты. В состоянии корневого компонента <App> останутся только свойства contacts и filter.

state = {  
 contacts: [],  
 filter: ''  
}

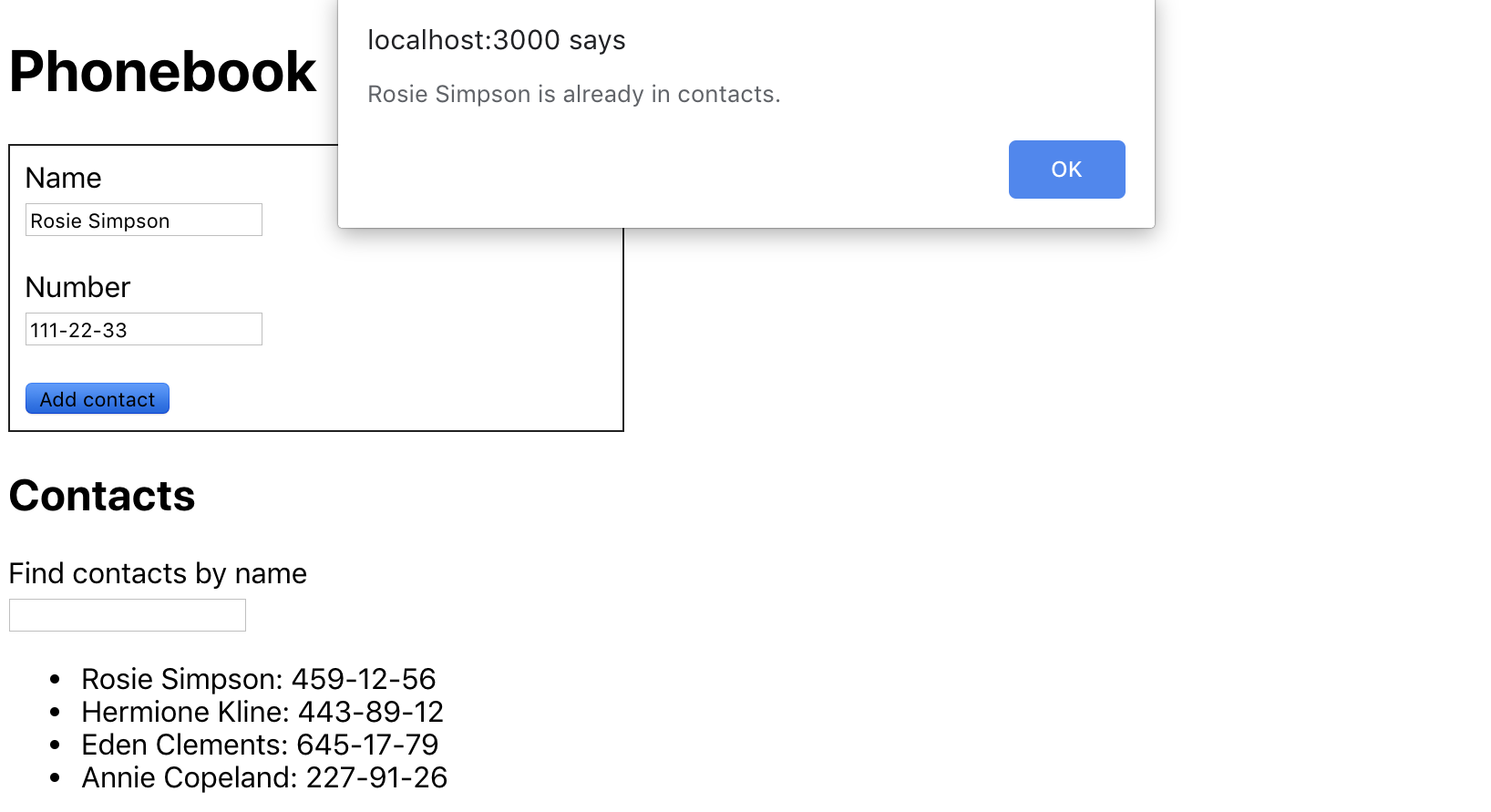
Достаточно выделить четыре компонента: форма добавления контактов, список контактов, элемент списка контактов и фильтр поиска.

После рефакторинга корневой компонент приложения будет выглядеть так.

<div>  
 <h1>Phonebook</h1>  
 <ContactForm ... />  
  
 <h2>Contacts</h2>  
 <Filter ... />  
 <ContactList ... />  
</div>

**Шаг 5**[**​**](https://textbook.edu.goit.global/lms-react-homework/v1/docs/hw-02/#%D1%88%D0%B0%D0%B3-5)

Запрети пользователю возможность добавлять контакты, имена которых уже есть в телефонной книге. При попытке выполнить такое действие выведи alert с предупреждением.



**Шаг 6**[**​**](https://textbook.edu.goit.global/lms-react-homework/v1/docs/hw-02/#%D1%88%D0%B0%D0%B3-6)

Расширь функционал приложения, позволив пользователю удалять ранее сохраненные контакты.

