В рамках данной лабораторной работы был рассмотрен React в качестве “View Layer”. Как описано в главе “React is just the view layer”, React исключительно сфокусирован на отображении UI. То есть его задача – преобразовать данные, которые я ему задаю в визуальные элементы. Разработчик в данном случае не взаимодействует с DOM напрямую, а описывает интерфейс и то, как он должен выглядеть в определенном состоянии.

В моей лабораторной работе был использован useState для создания счетчика. Когда пользователь нажимают одну из предложенных кнопок: increment или Decrement, то происходит вызов функции, которая обновляет состояние. Это показывает, что React видит, что данные изменились и запускает re-rendering, после чего компонент вызывается по новой и создает новое описание интерфейса с обновленным значением счетчика.

Как было выявлено ранее “Performance matters”. Поэтому главным пунктом в реализации данного интерфейса является производительность. React использует virtual DOM, что не создает нагрузки на систему. При каждом обновлении React создает новый virtual DOM и сравнивает его с прошлым, тем самым осуществляя diffing. После этого React выполняет patching – обновление только тех элементов, которые изменились. В итоге приложение остается максимально быстрым и производительным, так как нет необходимости перерисовывать страницу целиком. Таким образом, React автоматизирует сложный процесс по синхронизации пользовательского интерфейса, при этом не прибегая к сложным вычислениям и не нагружая само приложение.