МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА М.Д.МИЛЛИОНЩИКОВА»

Информационные системы в экономике
(наименование кафедры)

УТВЕРЖДЕН

на заседании кафедры
«02» 09 2026 г., протокол № 1

Заведующий кафедрой
Л.Р. Магомаева
(подпись)

ФОНД ОЦЕНОЧНЫХ СРЕДСТВОВ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

Разработка веб-приложений с применением фреймворков
(наименование дисциплины)

Направление /специальность подготовки
09.03.03 Прикладная информатика
(код и наименование направления/ специальности подготовки)

Направленность (профиль)
Разработка мобильных и Web-приложений
(наименование специализации / профиля подготовки)

Квалификация
бакалавр
(специалист / бакалавр / магистр)

Составитель М.К. Абдулаев
(подпись)

Грозный – 2026

ПАСПОРТ
ФОНДА ОЦЕНОЧНЫХ СРЕДСТВ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ
Разработка веб-приложений с применением фреймворков

№ п/п	Контролируемые разделы (темы) дисциплины	Код контролируемой компетенции (или ее части)	Наименование оценочного средства
6-й семестр			
1	**Тема 1.** Введение в разработку веб-приложений с использованием React.	(ОПК-2)	Лабораторная работа
2	**Тема 2.** Основы работы с React.: компоненты, JSX, Virtual DOM	(ПК-5)	Лабораторная работа
3	**Тема 3.** Управление состоянием в React.: useState и useEffect	(ОПК-2)	Лабораторная работа
4	**Тема 4.** Роутинг и навигация в веб-приложениях на React.	(ПК-5)	Лабораторная работа
5	**Тема 5.** Стилизация компонентов в React.: CSS Modules, Styled Components	(ОПК-2)	Лабораторная работа
6	**Тема 6.** Формы и обработка событий в React.	(ПК-5)	Лабораторная работа
7	**Тема 7.** Взаимодействие с сервером: AJAX запросы и работа с API	(ОПК-2)	Лабораторная работа
8	**Тема 8.** Аутентификация и авторизация в приложениях на React.	(ПК-5)	Лабораторная работа
7-й семестр			
9	**Тема 9.** Оптимизация производительности веб-приложений на React.	(ОПК-2)	Лабораторная работа

10	**Тема 10.** Тестирование компонентов и приложений на React.	(ПК-5)	Лабораторная работа
11	**Тема 11.** Разработка Single Page Applications (SPA) на React.	(ОПК-2)	Лабораторная работа
12	**Тема 12.** Работа с контекстом и хуками в React.	(ПК-5)	Лабораторная работа
13	**Тема 13**. Разработка Single Page Applications (SPA) на React.	(ОПК-2)	Лабораторная работа
14	**Тема 14.** Работа с контекстом и хуками в React.	(ПК-5)	Лабораторная работа

ПЕРЕЧЕНЬ ОЦЕНОЧНЫХ СРЕДСТВ

№ п/п	Наименование оценочного средства	Краткая характеристика оценочного средства	Представление оценочного средства в фонде
1	*Лабораторная работа*	Средство проверки умений применять полученные знания по заранее определенной методике для решения задач или заданий по модулю или дисциплине в целом	Комплект заданий для выполнения лабораторных работ
2	*Рубежный контроль*	Форма проверки знаний по дисциплине в виде первой и второй рубежных аттестаций	Вопросы к аттестациям
3	*зачет*	Итоговая форма оценки знаний	Вопросы к зачету
4	*экзамен*	Итоговая форма оценки знаний	Вопросы к экзамену

ЗАДАНИЯ ДЛЯ ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ

В качестве оценочных средств используются средства контроля выполнения лабораторных работ по дисциплине. Защита лабораторной работы – ответ на контрольные вопросы после выполнения лабораторной работы.

Средства текущего контроля: устный опрос (собеседование/опрос, разбор учебной ситуации на выбранную тему, подготовка устных сообщений и докладов), практическое задание (выполнение заданий в электронной форме на ПК).

6-й семестр

Лабораторная работа 1. Основы работы с React.	*Цель лабораторной работы:* Ознакомиться с базовыми принципами работы с библиотекой React, научиться создавать простые компоненты и использовать JSX синтаксис. (ОПК-2), (ПК-5)
Лабораторная работа 2. Управление состоянием в React.	*Цель лабораторной работы:* Изучить методы управления состоянием компонентов в React с помощью хуков useState и useEffect, научиться обновлять состояние и реагировать на изменения. (ОПК-2), (ПК-5)
Лабораторная работа 3. Роутинг и навигация в React.	*Цель лабораторной работы:* Познакомиться с библиотекой React Router и научиться создавать маршруты для навигации между страницами в веб-приложении на React. (ОПК-2), (ПК-5)
Лабораторная работа 4. Стилизация компонентов в React.	*Цель лабораторной работы* Изучить различные способы стилизации компонентов в React, включая CSS Modules и Styled Components, научиться создавать стилизованные интерфейсы. (ОПК-2), (ПК-5)
Лабораторная работа 5. Формы и обработка событий в React.	*Цель лабораторной работы:* Практически изучить создание и валидацию форм в React, научиться обрабатывать события в компонентах для реализации интерактивности. (ОПК-2), (ПК-5)

Лабораторная работа 6. Взаимодействие с сервером в React.	*Цель лабораторной работы:* Освоить основы взаимодействия с сервером в React-приложениях с использованием Fetch API для отправки запросов на сервер и получения данных. (ОПК-2), (ПК-5)
7-семестр	
Лабораторная работа 7. Оптимизация производительности в React.	*Цель лабораторной работы* Изучить методы оптимизации производительности React-приложений, включая мемоизацию, useMemo и useCallback, и применить их на практике. (ОПК-2), (ПК-5)
Лабораторная работа 8. Тестирование компонентов и приложений на React.	*Цель лабораторной работы:* Научиться писать модульные и интеграционные тесты для React-компонентов и приложений с использованием библиотеки Jest. (ОПК-2), (ПК-5)
Лабораторная работа 9. Разработка SPA на React.	*Цель лабораторной работы:* Практически освоить разработку Single Page Applications (SPA) на React, организовать структуру приложения и управлять его состоянием. (ОПК-2), (ПК-5)
Лабораторная работа 10. Работа с контекстом и хуками в React.	*Цель лабораторной работы* Изучить контекст и пользовательские хуки в React и применить их для передачи данных между компонентами и управления состоянием. (ОПК-2), (ПК-5)
Лабораторная работа 11. Архитектура приложений на React.	*Цель лабораторной работы* Разработать приложение с использованием лучших практик и паттернов, организовать его архитектуру и структуру компонентов. (ОПК-2), (ПК-5)
Лабораторная работа 12. SSR и применение в React.	*Цель лабораторной работы:* Изучить Server-Side Rendering (SSR) и его применение в React-приложениях для улучшения производительности и SEO. (ОПК-2), (ПК-5)

Критерии оценки ответов на лабораторные работы (7-й семестр)

Регламентом БРС предусмотрено всего 15 баллов за текущую работу студента. Критерии оценки разработаны, исходя из возможности ответа студентом до 6 лабораторных работ с использованием дополнительного материала по ним. (по 2 балла).

2 балла ставится за работу, выполненную полностью без ошибок и недочетов.

1 балл ставится за работу, выполненную полностью, но при наличии в ней негрубых ошибок и недочетов.

0 баллов ставится, если студент совсем не выполнил ни одного задания.

Максимальное количество баллов за выполнение и защиту лабораторных работ 12.

Максимально высокие баллы за текущий контроль – 15. За активное участие на лекциях и использование дополнительного материала при подготовке к занятиям студент получает дополнительно до 3 баллов.

Критерии оценки ответов на лабораторные работы (8-й семестр)

Регламентом БРС предусмотрено всего 15 баллов за текущую работу студента. Критерии оценки разработаны, исходя из возможности ответа студентом до 5 лабораторных работ с использованием дополнительного материала по ним. (по 2 балла).

3 балла ставится за работу, выполненную полностью без ошибок и недочетов.

2 балл ставится за работу, выполненную полностью, но при наличии в ней негрубых ошибок и недочетов.

0 баллов ставится, если студент совсем не выполнил ни одного задания.

Максимальное количество баллов за выполнение и защиту лабораторных работ 12.

Максимально высокие баллы за текущий контроль – 15. За активное участие на лекциях и использование дополнительного материала при подготовке к занятиям студент получает дополнительно до 3 баллов.

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА М.Д.МИЛЛИОНЩИКОВА

Институт цифровой экономики и технологического предпринимательства
Кафедра «Информационные системы в экономике»

Вопросы к первой рубежной аттестации «Разработка веб-приложений с применением фреймворков» (6-й семестр)

1. Какова основная цель использования библиотеки React в разработке веб-приложений?
2. Какие основные преимущества предоставляет JSX в сравнении с обычным JavaScript?
3. Какие методы React используются для управления состоянием компонентов? Приведите примеры их применения.
4. Как осуществляется роутинг и навигация в веб-приложениях на React? Какие инструменты для этого используются?
5. Как происходит стилизация компонентов в React-приложениях? Расскажите о преимуществах подходов CSS Modules и Styled Components.
6. Как обрабатываются формы и события в React-приложениях? Как организовать валидацию форм?
7. Как осуществляется взаимодействие с сервером в React-приложениях? Укажите на основные методы и инструменты.
8. Какие механизмы используются для реализации аутентификации и авторизации в приложениях на React?
9. Какие средства и методы оптимизации производительности могут быть применены в React-приложениях?
10. Какие инструменты и подходы используются для тестирования компонентов и приложений на React?

Вопросы ко второй рубежной аттестации «Разработка веб-приложений с применением фреймворков» (7-й семестр)

1. Каковы особенности разработки Single Page Applications (SPA) на React? В чем их преимущества и недостатки?
2. Какие методы и инструменты используются для работы с контекстом и хуками в React-приложениях?
3. Какие паттерны и лучшие практики применяются при проектировании архитектуры приложений на React?
4. Как SSR (Server-Side Rendering) влияет на производительность веб-приложений на React? В каких случаях его стоит использовать?

5. Что такое Next.js и какие возможности он предоставляет для разработки приложений на React?
6. Какие средства и методы используются для тестирования компонентов и приложений на React?
7. Какие аспекты безопасности следует учитывать при разработке и аутентификации в приложениях на React?
8. Какие сценарии оптимизации производительности чаще всего применяются в веб-приложениях на React?
9. Какие особенности работы с API следует учитывать при взаимодействии с сервером в React-приложениях?
10. Какие инструменты и методы используются для управления состоянием приложения в React и обеспечения его целостности?

Критерии оценки ответов на рубежной аттестации

Регламентом БРС предусмотрено всего 20 баллов за рубежную аттестацию студента. Критерии оценки разработаны, исходя из возможности ответа студентом на 2 вопроса в билете (по 10 баллов).

10 баллов (5+) заслуживает студент, обнаруживший всестороннее, систематическое и глубокое знание учебного программного материала, самостоятельно выполнивший все предусмотренные программой задания, глубоко усвоивший основную и дополнительную литературу, рекомендованную программой, активно работавший на практических, семинарских, лабораторных занятиях, разбирающийся в основных научных концепциях по изучаемой дисциплине, проявивший творческие способности и научный подход в понимании и изложении учебного программного материала, ответ отличается богатством и точностью использованных терминов, материал излагается последовательно и логично.

9 баллов (5) заслуживает студент, обнаруживший всестороннее, систематическое знание учебного программного материала, самостоятельно выполнивший все предусмотренные программой задания, глубоко усвоивший основную литературу и знаком с дополнительной литературой, рекомендованной программой, активно работавший на практических, семинарских, лабораторных занятиях, показавший систематический характер знаний по дисциплине, достаточный для дальнейшей учебы, а также способность к их самостоятельному пополнению, ответ отличается точностью использованных терминов, материал излагается последовательно и логично.

8 баллов (4+) заслуживает студент, обнаруживший полное знание учебно-программного материала, не допускающий в ответе существенных неточностей, самостоятельно выполнивший все предусмотренные программой задания, усвоивший основную литературу, рекомендованную программой, активно работавший на практических, семинарских, лабораторных занятиях, показавший систематический характер знаний по дисциплине, достаточный для дальнейшей учебы, а также способность к их самостоятельному пополнению.

7 баллов (4) заслуживает студент, обнаруживший достаточно полное знание учебно-программного материала, не допускающий в ответе существенных неточностей, самостоятельно выполнивший все предусмотренные программой задания, усвоивший основную литературу, рекомендованную программой, активно работавший на практических, семинарских, лабораторных занятиях, показавший систематический характер знаний по дисциплине, достаточный для дальнейшей учебы, а также способность к их самостоятельному пополнению.

6 баллов (4-) заслуживает студент, обнаруживший достаточно полное знание учебно-программного материала, не допускающий в ответе существенных неточностей, самостоятельно выполнивший основные предусмотренные программой задания, усвоивший основную литературу, рекомендованную программой, отличавшийся достаточной

активностью на практических (семинарских) и лабораторных занятиях, показавший систематический характер знаний по дисциплине, достаточный для дальнейшей учебы.

5 баллов (3+) заслуживает студент, обнаруживший знание основного учебно-программного материала в объёме, необходимом для дальнейшей учебы и предстоящей работы по профессии, не отличавшийся активностью на практических (семинарских) и лабораторных занятиях, самостоятельно выполнивший основные предусмотренные программой задания, усвоивший основную литературу, рекомендованную программой, однако допустивший некоторые погрешности при их выполнении и в ответе па экзамене, но обладающий необходимыми знаниями для их самостоятельного устранения.

4 балла (3) заслуживает студент, обнаруживший знание основного учебно-программного материала в объёме, необходимом для дальнейшей учебы и предстоящей работы по профессии, не отличавшийся активностью на практических (семинарских) и лабораторных занятиях, самостоятельно выполнивший основные предусмотренные программой задания, усвоивший основную литературу, рекомендованную программой, однако допустивший некоторые погрешности при их выполнении и в ответе на экзамене, но обладающий необходимыми знаниями для устранения под руководством преподавателя допущенных погрешностей.

3 балла (3-) заслуживает студент, обнаруживший знание основного учебно-программного материала в объёме, необходимом для дальнейшей учебы и предстоящей работы по профессии, не отличавшийся активностью на практических (семинарских) и лабораторных занятиях, самостоятельно выполнивший основные предусмотренные программой задания, однако допустивший погрешности при их выполнении и в ответе на экзамене, но обладающий необходимыми знаниями для устранения под руководством преподавателя наиболее существенных погрешностей.

2 балла (2) выставляется студенту, обнаружившему пробелы в знаниях или отсутствие знаний по значительной части основного учебно-программного материала, не выполнившему самостоятельно предусмотренные программой основные задания, допустившему принципиальные ошибки в выполнении предусмотренных программой заданий, не отработавшему основные практические, семинарские, лабораторные занятия, допускающему существенные ошибки при ответе, и который не может продолжить обучение или приступить к профессиональной деятельности без дополнительных занятий по соответствующей дисциплине.

1 балл — нет ответа (отказ от ответа, представленный ответ полностью не по существу содержащихся в экзаменационном задании вопросов)

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА М.Д.МИЛЛИОНЩИКОВА

Институт цифровой экономики и технологического предпринимательства
Кафедра информационные системы в экономике

Вопросы к зачету по дисциплине «Разработка веб-приложений с применением фреймворков» (ОПК-2), (ПК-5)

1. Какова основная цель использования библиотеки React в разработке веб-приложений?
2. Какие основные преимущества предоставляет JSX в сравнении с обычным JavaScript? (ОПК-2), (ПК-5)
3. Какие методы React используются для управления состоянием компонентов? Приведите примеры их применения. (ОПК-2), (ПК-5)
4. Как осуществляется роутинг и навигация в веб-приложениях на React? Какие инструменты для этого используются? (ОПК-2), (ПК-5)
5. Как происходит стилизация компонентов в React-приложениях? Расскажите о преимуществах подходов CSS Modules и Styled Components.
6. Как обрабатываются формы и события в React-приложениях? Как организовать валидацию форм?
7. Как осуществляется взаимодействие с сервером в React-приложениях? Укажите на основные методы и инструменты. (ОПК-2), (ПК-5)
8. Какие механизмы используются для реализации аутентификации и авторизации в приложениях на React?
9. Какие средства и методы оптимизации производительности могут быть применены в React-приложениях?
10. Какие инструменты и подходы используются для тестирования компонентов и приложений на React? (ОПК-2), (ПК-5)
11. Каковы особенности разработки Single Page Applications (SPA) на React? В чем их преимущества и недостатки?
12. Какие методы и инструменты используются для работы с контекстом и хуками в React-приложениях?
13. Какие паттерны и лучшие практики применяются при проектировании архитектуры приложений на React? (ОПК-2), (ПК-5)
14. Как SSR (Server-Side Rendering) влияет на производительность веб-приложений на React? В каких случаях его стоит использовать?

15. Что такое Next.js и какие возможности он предоставляет для разработки приложений на React?

16. Какие средства и методы используются для тестирования компонентов и приложений на React?

17. Какие аспекты безопасности следует учитывать при разработке и аутентификации в приложениях на React?

18. Какие сценарии оптимизации производительности чаще всего применяются в веб-приложениях на React? (ОПК-2), (ПК-5)

19. Какие особенности работы с API следует учитывать при взаимодействии с сервером в React-приложениях?

20. Какие инструменты и методы используются для управления состоянием приложения в React и обеспечения его целостности?

Критерии оценки ответов на зачете

Регламентом БРС предусмотрено 20 баллов (максимальный балл) за ответ на вопросы в билете. Критерии оценки разработаны, исходя из возможности ответа студентом на 4 вопроса в билете (по 5 баллов).

5 баллов - Дан полный, развернутый ответ на поставленный вопрос, показана совокупность осознанных знаний по дисциплине, доказательно раскрыты основные положения вопросов; в ответе прослеживается четкая структура, логическая последовательность, отражающая сущность раскрываемых понятий, теорий, явлений. Могут быть допущены недочеты в определении понятий, исправленные студентом самостоятельно в процессе ответа.

4 балла - Дан полный, развернутый ответ на поставленный вопрос, показано умение выделить существенные и несущественные признаки, причинно-следственные связи. Ответ четко структурирован, логичен, изложен литературным языком с использованием современной технической терминологии. Могут быть допущены некоторые неточности или незначительные ошибки, исправленные студентом с помощью преподавателя.

3 балла - Дан недостаточно полный и недостаточно развернутый ответ. Логика и последовательность изложения имеют нарушения. Допущены ошибки в раскрытии понятий, употреблении терминов. Студент не способен самостоятельно выделить существенные и несущественные признаки и причинно-следственные связи. В ответе отсутствуют выводы. Умение раскрыть значение обобщенных знаний не показано. Речевое оформление требует поправок, коррекции.

2 балла - Ответ представляет собой разрозненные знания с существенными ошибками по вопросу. Присутствуют фрагментарность, нелогичность изложения. Студент осознает связь обсуждаемого вопроса по билету с другими объектами дисциплины.

1 балл - Отсутствуют выводы, конкретизация и доказательность изложения. Речь неграмотная, техническая терминология не используется. Дополнительные и уточняющие вопросы преподавателя приводят к незначительной коррекции ответа студента.

0 баллов - Ответ на вопрос полностью отсутствует, либо отказ от ответа.

Критерии оценки знаний студента на зачете

Оценка «зачтено» - выставляется студенту, показавшему фрагментарный, разрозненный характер знаний, варьируемых от оценки «отлично» до «удовлетворительно». При этом он владеет основными разделами учебной программы, необходимыми для дальнейшего обучения и может применять полученные знания по образцу в стандартной ситуации.

Оценка «не зачтено» - выставляется студенту, который не знает большей части основного содержания учебной программы дисциплины, допускает грубые ошибки в формулировках основных понятий дисциплины и не умеет использовать полученные знания при решении типовых практических задач.

Вопросы к первой рубежной аттестации (7-й семестр)

1. Какие основные принципы оптимизации производительности следует учитывать при разработке веб-приложений на React?
2. Какие инструменты и методы можно использовать для оценки производительности веб-приложений на React?
3. Какие типичные проблемы производительности могут возникать в веб-приложениях на React и как их можно решить?
4. Какие сценарии тестирования компонентов и приложений на React вы считаете наиболее важными для обеспечения качества кода?
5. Какие основные методы и инструменты используются для тестирования компонентов и приложений на React?
6. Какие преимущества и недостатки связаны с разработкой Single Page Applications (SPA) на React?
7. Какие особенности разработки SPA на React следует учитывать для обеспечения быстрой загрузки и отзывчивости приложения?
8. Как работает контекст в React и какие задачи можно решать с его помощью? Приведите примеры использования контекста.
9. Какие хуки доступны в React и для каких задач они предназначены? Приведите примеры использования хуков.
10. Какие паттерны и лучшие практики применяются при проектировании архитектуры приложений на React?

Вопросы ко второй рубежной аттестации (7-й семестр)

1. Как SSR (Server-Side Rendering) влияет на производительность и SEO веб-приложений на React? В каких случаях его стоит применять?
2. Какие основные преимущества предоставляет SSR в сравнении с клиентским рендерингом?
3. Какие основные принципы и подходы следует учитывать при разработке архитектуры приложений на React с использованием SSR?
4. Какие ключевые особенности и возможности предоставляет фреймворк Next.js для разработки веб-приложений на React?
5. Какие основные задачи можно решать с помощью Next.js и какие преимущества он предоставляет разработчикам?
6. Какие сценарии тестирования компонентов и приложений на React вы считаете наиболее важными для обеспечения качества кода?
7. Какие средства и методы используются для тестирования компонентов и приложений на React?
8. Какие архитектурные паттерны и лучшие практики помогают обеспечить масштабируемость и поддерживаемость приложений на React?
9. Какие сценарии оптимизации производительности чаще всего применяются в веб-приложениях на React?
10. Какие аспекты безопасности следует учитывать при разработке приложений на React с использованием SSR и Next.js?

Критерии оценки ответов на рубежной аттестации

Регламентом БРС предусмотрено всего 20 баллов за рубежную аттестацию студента. Критерии оценки разработаны, исходя из возможности ответа студентом на 2 вопроса в билете (по 10 баллов).

10 баллов (5+) заслуживает студент, обнаруживший всестороннее, систематическое и глубокое знание учебного программного материала, самостоятельно выполнивший все предусмотренные программой задания, глубоко усвоивший основную и дополнительную литературу, рекомендованную программой, активно работавший на практических, семинарских, лабораторных занятиях, разбирающийся в основных научных концепциях по изучаемой дисциплине, проявивший творческие способности и научный подход в понимании и изложении учебного программного материала, ответ отличается богатством и точностью использованных терминов, материал излагается последовательно и логично.

9 баллов (5) заслуживает студент, обнаруживший всестороннее, систематическое знание учебного программного материала, самостоятельно выполнивший все предусмотренные программой задания, глубоко усвоивший основную литературу и знаком с дополнительной литературой, рекомендованной программой, активно работавший на практических, семинарских, лабораторных занятиях, показавший систематический характер знаний по дисциплине, достаточный для дальнейшей учебы, а также способность к их самостоятельному пополнению, ответ отличается точностью использованных терминов, материал излагается последовательно и логично.

8 баллов (4+) заслуживает студент, обнаруживший полное знание учебно-программного материала, не допускающий в ответе существенных неточностей, самостоятельно выполнивший все предусмотренные программой задания, усвоивший основную литературу, рекомендованную программой, активно работавший на практических, семинарских, лабораторных занятиях, показавший систематический характер знаний по дисциплине, достаточный для дальнейшей учебы, а также способность к их самостоятельному пополнению.

7 баллов (4) заслуживает студент, обнаруживший достаточно полное знание учебно-программного материала, не допускающий в ответе существенных неточностей, самостоятельно выполнивший все предусмотренные программой задания, усвоивший основную литературу, рекомендованную программой, активно работавший на практических, семинарских, лабораторных занятиях, показавший систематический характер знаний по дисциплине, достаточный для дальнейшей учебы, а также способность к их самостоятельному пополнению.

6 баллов (4-) заслуживает студент, обнаруживший достаточно полное знание учебно-программного материала, не допускающий в ответе существенных неточностей, самостоятельно выполнивший основные предусмотренные программой задания, усвоивший основную литературу, рекомендованную программой, отличавшийся достаточной

активностью на практических (семинарских) и лабораторных занятиях, показавший систематический характер знаний по дисциплине, достаточный для дальнейшей учебы.

5 баллов (3+) заслуживает студент, обнаруживший знание основного учебно-программного материала в объёме, необходимом для дальнейшей учебы и предстоящей работы по профессии, не отличавшийся активностью на практических (семинарских) и лабораторных занятиях, самостоятельно выполнивший основные предусмотренные программой задания, усвоивший основную литературу, рекомендованную программой, однако допустивший некоторые погрешности при их выполнении и в ответе па экзамене, но обладающий необходимыми знаниями для их самостоятельного устранения.

4 балла (3) заслуживает студент, обнаруживший знание основного учебно-программного материала в объёме, необходимом для дальнейшей учебы и предстоящей работы по профессии, не отличавшийся активностью на практических (семинарских) и лабораторных занятиях, самостоятельно выполнивший основные предусмотренные программой задания, усвоивший основную литературу, рекомендованную программой, однако допустивший некоторые погрешности при их выполнении и в ответе на экзамене, но обладающий необходимыми знаниями для устранения под руководством преподавателя допущенных погрешностей.

3 балла (3-) заслуживает студент, обнаруживший знание основного учебно-программного материала в объёме, необходимом для дальнейшей учебы и предстоящей работы по профессии, не отличавшийся активностью на практических (семинарских) и лабораторных занятиях, самостоятельно выполнивший основные предусмотренные программой задания, однако допустивший погрешности при их выполнении и в ответе на экзамене, но обладающий необходимыми знаниями для устранения под руководством преподавателя наиболее существенных погрешностей.

2 балла (2) выставляется студенту, обнаружившему пробелы в знаниях или отсутствие знаний по значительной части основного учебно-программного материала, не выполнившему самостоятельно предусмотренные программой основные задания, допустившему принципиальные ошибки в выполнении предусмотренных программой заданий, не отработавшему основные практические, семинарские, лабораторные занятия, допускающему существенные ошибки при ответе, и который не может продолжить обучение или приступить к профессиональной деятельности без дополнительных занятий по соответствующей дисциплине.

1 балл — нет ответа (отказ от ответа, представленный ответ полностью не по существу содержащихся в экзаменационном задании вопросов)

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА М.Д.МИЛЛИОНЩИКОВА

Институт цифровой экономики и технологического предпринимательства
Кафедра информационные системы в экономике

Вопросы к экзамену по дисциплине «Разработка веб-приложений с применением фреймворков» (7-й семестр)

1. Какие основные принципы оптимизации производительности следует учитывать при разработке веб-приложений на React?
2. Какие инструменты и методы можно использовать для оценки производительности веб-приложений на React?
3. Какие типичные проблемы производительности могут возникать в веб-приложениях на React и как их можно решить?
4. Какие сценарии тестирования компонентов и приложений на React вы считаете наиболее важными для обеспечения качества кода?
5. Какие основные методы и инструменты используются для тестирования компонентов и приложений на React?
6. Какие преимущества и недостатки связаны с разработкой Single Page Applications (SPA) на React? (ОПК-2), (ПК-5)
7. Какие особенности разработки SPA на React следует учитывать для обеспечения быстрой загрузки и отзывчивости приложения?
8. Как работает контекст в React и какие задачи можно решать с его помощью? Приведите примеры использования контекста.
9. Какие хуки доступны в React и для каких задач они предназначены? Приведите примеры использования хуков.
10. Какие паттерны и лучшие практики применяются при проектировании архитектуры приложений на React?
11. Как SSR (Server-Side Rendering) влияет на производительность и SEO веб-приложений на React? В каких случаях его стоит применять?
12. Какие основные преимущества предоставляет SSR в сравнении с клиентским рендерингом?
13. Какие основные принципы и подходы следует учитывать при разработке архитектуры приложений на React с использованием SSR?
14. Какие ключевые особенности и возможности предоставляет фреймворк Next.js для разработки веб-приложений на React? (ОПК-2), (ПК-5)
15. Какие основные задачи можно решать с помощью Next.js и какие преимущества он предоставляет разработчикам?
16. Какие сценарии тестирования компонентов и приложений на React вы считаете наиболее важными для обеспечения качества кода?
17. Какие средства и методы используются для тестирования компонентов и приложений на React? (ОПК-2), (ПК-5)

18. Какие архитектурные паттерны и лучшие практики помогают обеспечить масштабируемость и поддерживаемость приложений на React?
19. Какие сценарии оптимизации производительности чаще всего применяются в веб-приложениях на React?
20. Какие аспекты безопасности следует учитывать при разработке приложений на React с использованием SSR и Next.js?

Критерии оценки ответов на экзамене

Регламентом БРС предусмотрено всего 20 баллов за экзамен. Критерии оценки разработаны, исходя из возможности ответа студентом на 2 вопроса в билете (по 10 баллов).

10 баллов (5+) заслуживает студент, обнаруживший всестороннее, систематическое и глубокое знание учебного программного материала, самостоятельно выполнивший все предусмотренные программой задания, глубоко усвоивший основную и дополнительную литературу, рекомендованную программой, активно работавший на практических, семинарских, лабораторных занятиях, разбирающийся в основных научных концепциях по изучаемой дисциплине, проявивший творческие способности и научный подход в понимании и изложении учебного программного материала, ответ отличается богатством и точностью использованных терминов, материал излагается последовательно и логично.

9 баллов (5) заслуживает студент, обнаруживший всестороннее, систематическое знание учебного программного материала, самостоятельно выполнивший все предусмотренные программой задания, глубоко усвоивший основную литературу и знаком с дополнительной литературой, рекомендованной программой, активно работавший на практических, семинарских, лабораторных занятиях, показавший систематический характер знаний по дисциплине, достаточный для дальнейшей учебы, а также способность к их самостоятельному пополнению, ответ отличается точностью использованных терминов, материал излагается последовательно и логично.

8 баллов (4+) заслуживает студент, обнаруживший полное знание учебно-программного материала, не допускающий в ответе существенных неточностей, самостоятельно выполнивший все предусмотренные программой задания, усвоивший основную литературу, рекомендованную программой, активно работавший на практических, семинарских, лабораторных занятиях, показавший систематический характер знаний по дисциплине, достаточный для дальнейшей учебы, а также способность к их самостоятельному пополнению.

7 баллов (4) заслуживает студент, обнаруживший достаточно полное знание учебно-программного материала, не допускающий в ответе существенных неточностей, самостоятельно выполнивший все предусмотренные программой задания, усвоивший основную литературу, рекомендованную программой, активно работавший на практических, семинарских, лабораторных занятиях, показавший систематический характер знаний по дисциплине, достаточный для дальнейшей учебы, а также способность к их самостоятельному пополнению.

6 баллов (4-) заслуживает студент, обнаруживший достаточно полное знание учебно-программного материала, не допускающий в ответе существенных неточностей, самостоятельно выполнивший основные предусмотренные программой задания, усвоивший основную литературу, рекомендованную программой, отличавшийся достаточной активностью на практических (семинарских) и лабораторных занятиях, показавший систематический характер знаний по дисциплине, достаточный для дальнейшей учебы.

5 баллов (3+) заслуживает студент, обнаруживший знание основного учебно-программного материала в объёме, необходимом для дальнейшей учебы и предстоящей работы по профессии, не отличавшийся активностью на практических (семинарских) и лабораторных занятиях, самостоятельно выполнивший основные предусмотренные программой задания, усвоивший основную литературу, рекомендованную программой, однако допустивший некоторые погрешности при их выполнении и в ответе па экзамене, но обладающий необходимыми знаниями для их самостоятельного устранения.

4 балла (3) заслуживает студент, обнаруживший знание основного учебно-программного материала в объёме, необходимом для дальнейшей учебы и предстоящей работы по профессии, не отличавшийся активностью на практических (семинарских) и лабораторных занятиях, самостоятельно выполнивший основные предусмотренные программой задания, усвоивший основную литературу, рекомендованную программой, однако допустивший некоторые погрешности при их выполнении и в ответе на экзамене, но обладающий необходимыми знаниями для устранения под руководством преподавателя допущенных погрешностей.

3 балла (3-) заслуживает студент, обнаруживший знание основного учебно-программного материала в объёме, необходимом для дальнейшей учебы и предстоящей работы по профессии, не отличавшийся активностью на практических (семинарских) и лабораторных занятиях, самостоятельно выполнивший основные предусмотренные программой задания, однако допустивший погрешности при их выполнении и в ответе на экзамене, но обладающий необходимыми знаниями для устранения под руководством преподавателя наиболее существенных погрешностей.

2 балла (2) выставляется студенту, обнаружившему пробелы в знаниях или отсутствие знаний по значительной части основного учебно-программного материала, не выполнившему самостоятельно предусмотренные программой основные задания, допустившему принципиальные ошибки в выполнении предусмотренных программой заданий, не отработавшему основные практические, семинарские, лабораторные занятия, допускающему существенные ошибки при ответе, и который не может продолжить обучение или приступить к профессиональной деятельности без дополнительных занятий по соответствующей дисциплине.

1 балл — нет ответа (отказ от ответа, представленный ответ полностью не по существу содержащихся в экзаменационном задании вопросов)

Критерии оценки ответов на экзамене

Оценку "отлично" заслуживает студент, обнаруживший всестороннее, систематическое и глубокое знание учебно-программного материала, умение свободно выполнять задания, предусмотренные программой, усвоивший основную и знакомый с дополнительной литературой, рекомендованной программой. Как правило, оценка "отлично" выставляется студентам, усвоившим взаимосвязь основных понятий дисциплины в их значении для приобретаемой профессии, проявившим творческие способности в понимании, изложении и использовании учебно-программного материала.

Оценку "хорошо" заслуживает студент, обнаруживший полное знание учебно-программного материала, успешно выполняющий предусмотренные в программе задания, усвоивший основную литературу, рекомендованную в программе. Как правило, оценка "хорошо" выставляется студентам, показавшим систематический характер знаний по дисциплине и способным к их самостоятельному пополнению и обновлению в ходе дальнейшей учебной работы и профессиональной деятельности.

Оценку "удовлетворительно" заслуживает студент, обнаруживший знания основного учебно-программного материала в объеме, необходимом для дальнейшей учебы и предстоящей работы по специальности, справляющийся с выполнением заданий, предусмотренных программой, знакомый с основной литературой, рекомендованной программой. Как правило, оценка "удовлетворительно" выставляется студентам, допустившим погрешности в ответе на экзамене и при выполнении экзаменационных заданий, но обладающим необходимыми знаниями для их устранения под руководством преподавателя.

Оценка "неудовлетворительно" выставляется студенту, обнаружившему пробелы в знаниях основного учебно-программного материала, допустившему принципиальные ошибки в выполнении предусмотренных программой заданий. Как правило, оценка "неудовлетворительно" ставится студентам, которые не могут продолжить обучение или приступить к профессиональной деятельности по окончании вуза без дополнительных занятий по соответствующей дисциплине.

Приложение 1

КОМПЛЕКТ ЗАДАНИЙ ДЛЯ ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ

Лабораторная работа 1: Основы работы с React

Цель лабораторной работы:

Ознакомиться с базовыми принципами работы с библиотекой React, научиться создавать простые компоненты и использовать JSX синтаксис.

Описание работы:

1. **Установка и настройка проекта**
   - **Создание нового проекта:**
     - Убедитесь, что у вас установлены Node.js и npm.
     - В командной строке выполните команду для создания нового проекта с помощью Create React App:

```
npx create-react-app my-first-react-app
```

     - Перейдите в созданную директорию проекта:

```
cd my-first-react-app
```

     - Запустите проект:

```
npm start
```

     - В браузере откроется страница с приветственным сообщением от Create React App.
2. **Создание простого компонента**
   - **Создание компонента:**
     - Откройте проект в вашем любимом редакторе кода (например, VS Code).
     - В директории `src` создайте файл `HelloWorld.js`.
     - Добавьте в `HelloWorld.js` следующий код для создания простого компонента:

```
import React from 'react';

function HelloWorld() {
    return (
        <div>
            <h1>Hello, World!</h1>
        </div>
    );
}

export default HelloWorld;
```

- **Использование компонента:**
  - Откройте файл `App.js`.
  - Импортируйте компонент `HelloWorld`:

```
import React from 'react';
import HelloWorld from './HelloWorld';

function App() {
    return (
        <div className="App">
            <HelloWorld />
        </div>
    );
}

export default App;
```

3. **Использование JSX синтаксиса**
   - **Изучение JSX:**
     - В React компоненты пишутся с использованием JSX — синтаксиса, который позволяет писать HTML-подобный код внутри JavaScript.
     - В компоненте `HelloWorld` уже используется JSX для отображения заголовка `<h1>`.
   - **Добавление элементов с использованием JSX:**
     - Обновите компонент `HelloWorld`, чтобы он отображал дополнительные элементы:

```
import React from 'react';

function HelloWorld() {
    return (
        <div>
            <h1>Hello, World!</h1>
            <p>Welcome to your first React component.</p>
            <ul>
                <li>Learn JSX syntax</li>
                <li>Create React components</li>
                <li>Build amazing UIs</li>
            </ul>
        </div>
    );
}

export default HelloWorld;
```

4. **Стилизация компонентов**
    - **Добавление CSS:**
        - Создайте файл `HelloWorld.css` в директории `src`.
        - Добавьте стили для компонента `HelloWorld`:

```
.hello-world {
    text-align: center;
    color: #333;
}
```

    - **Применение стилей к компоненту:**
        - Обновите файл `HelloWorld.js` для использования CSS:

```
import React from 'react';
import './HelloWorld.css';

function HelloWorld() {
    return (
        <div className="hello-world">
            <h1>Hello, World!</h1>
            <p>Welcome to your first React component.</p>
            <ul>
                <li>Learn JSX syntax</li>
                <li>Create React components</li>
                <li>Build amazing UIs</li>
            </ul>
        </div>
    );
}

export default HelloWorld;
```

5. **Композиция компонентов**
    - **Создание нового компонента:**

- В директории `src` создайте файл `ListItem.js`.
- Добавьте в `ListItem.js` следующий код:

```
import React from 'react';

function ListItem(props) {
    return <li>{props.item}</li>;
}

export default ListItem;
```

- **Использование компонента `ListItem` в компоненте `HelloWorld`:**
  - Обновите файл `HelloWorld.js`:

```
import React from 'react';
import './HelloWorld.css';
import ListItem from './ListItem';

function HelloWorld() {
    const items = ['Learn JSX syntax', 'Create React
components', 'Build amazing UIs'];

    return (
        <div className="hello-world">
            <h1>Hello, World!</h1>
            <p>Welcome to your first React component.</p>
            <ul>
                {items.map((item, index) => (
                    <ListItem key={index} item={item} />
                ))}
            </ul>
        </div>
    );
}

export default HelloWorld;
```

6. **Работа с состоянием компонентов**
   - **Создание компонента с состоянием:**
     - В директории `src` создайте файл `Counter.js`.
     - Добавьте в `Counter.js` следующий код:

```
import React, { useState } from 'react';

function Counter() {
    const [count, setCount] = useState(0);

    return (
        <div>
            <p>Current count: {count}</p>
            <button onClick={() => setCount(count +
1)}>Increment</button>
            <button onClick={() => setCount(count -
1)}>Decrement</button>
        </div>
    );
}

export default Counter;
```

- **Использование компонента `Counter` в компоненте `HelloWorld`:**
  - Обновите файл `HelloWorld.js`:

```
import React from 'react';
import './HelloWorld.css';
import ListItem from './ListItem';
import Counter from './Counter';

function HelloWorld() {
    const items = ['Learn JSX syntax', 'Create React
components', 'Build amazing UIs'];

    return (
        <div className="hello-world">
            <h1>Hello, World!</h1>
            <p>Welcome to your first React component.</p>
            <ul>
                {items.map((item, index) => (
                    <ListItem key={index} item={item} />
                ))}
            </ul>
            <Counter />
        </div>
    );
}

export default HelloWorld;
```

7. **Работа с формами и обработкой событий**
   - **Создание компонента формы:**
     - В директории `src` создайте файл `Form.js`.
     - Добавьте в `Form.js` следующий код:

```
import React, { useState } from 'react';

function Form() {
    const [inputValue, setInputValue] = useState('');

    const handleSubmit = (event) => {
        event.preventDefault();
        alert(`Submitted value: ${inputValue}`);
    };

    return (
        <form onSubmit={handleSubmit}>
            <label>
                Input:
                <input
                    type="text"
                    value={inputValue}
                    onChange={(e) =>
setInputValue(e.target.value)}
                />
            </label>
            <button type="submit">Submit</button>
        </form>
    );
}

export default Form;
```

- **Использование компонента `Form` в компоненте `HelloWorld`:**
  - Обновите файл `HelloWorld.js`:

```
import React from 'react';
import './HelloWorld.css';
import ListItem from './ListItem';
import Counter from './Counter';
import Form from './Form';
function HelloWorld() {
    const items = ['Learn JSX syntax', 'Create React components',
'Build amazing UIs'];
    return (
        <div className="hello-world">
            <h1>Hello, World!</h1>
            <p>Welcome to your first React component.</p>
            <ul>
                {items.map((item, index) => (
                    <ListItem key={index} item={item} />
                ))}
            </ul>
            <Counter />
            <Form />
        </div>
    );
}

export default HelloWorld;
```

8. **Использование эффектов**
    - **Создание компонента с эффектом:**
        - В директории `src` создайте файл `DataFetcher.js`.
        - Добавьте в `DataFetcher.js` следующий код:

```
import React, { useState, useEffect } from 'react';

function DataFetcher() {
    const [data, setData] = useState(null);
    const [loading, setLoading] = useState(true);

    useEffect(() => {
        fetch('https://jsonplaceholder.typicode.com/posts/1')
            .then((response) => response.json())
            .then((data) => {
                setData(data);
                setLoading(false);
            });
    }, []);

    if (loading) {
        return <p>Loading...</p>;
    }

    return (
        <div>
            <h3>{data.title}</h3>
            <p>{data.body}</p>
        </div>
    );
}

export default DataFetcher;
```

    - **Использование компонента `DataFetcher` в компоненте `HelloWorld`:**
        - Обновите файл `HelloWorld.js`:

```
import React from 'react';
import './HelloWorld.css';
import ListItem from './ListItem';
import Counter from './Counter';
import Form from './Form';
import DataFetcher from './DataFetcher';

function HelloWorld() {
    const items = ['Learn JSX syntax', 'Create React components',
'Build amazing UIs'];

    return (
        <div className="hello-world">
            <h1>Hello, World!</h1>
            <p>Welcome to your first React component.</p>
            <ul>
                {items.map((item, index) => (
                    <ListItem key={index} item={item} />
                ))}
            </ul>
            <Counter />
            <Form />
            <DataFetcher />
        </div>
    );
}

export default HelloWorld;
```

9. **Реализация маршрутизации**
    - **Установка react-router-dom:**
        - В командной строке выполните команду:

```
npm install react-router-dom
```

    - **Настройка маршрутизации:**
        - Обновите файл `App.js`:

```
import React from 'react';
import { BrowserRouter as Router, Route, Switch, Link } from 'react-
router-dom';
import HelloWorld from './HelloWorld';
import Counter from './Counter';
import Form from './Form';
import DataFetcher from './DataFetcher';

function App() {
    return (
        <Router>
            <div className="App">
                <nav>
                    <ul>
                        <li>
                            <Link to="/">Home</Link>
                        </li>
                        <li>
                            <Link to="/counter">Counter</Link>
                        </li>
                        <li>
                            <Link to="/form">Form</Link>
                        </li>
                        <li>
                            <Link to="/data-fetcher">Data
Fetcher</Link>
                        </li>
                    </ul>
                </nav>
                <Switch>
                    <Route path="/" exact component={HelloWorld} />
                    <Route path="/counter" component={Counter} />
                    <Route path="/form" component={Form} />
                    <Route path="/data-fetcher"
component={DataFetcher} />
                </Switch>
            </div>
        </Router>
    );
}

export default App;
```

- **Создание отдельных компонентов для маршрутов:**
  - Обновите файл `HelloWorld.js` для использования в качестве домашней страницы:

```
import React from 'react';
import './HelloWorld.css';
import ListItem from './ListItem';

function HelloWorld() {
    const items = ['Learn JSX syntax', 'Create React components',
'Build amazing UIs'];

    return (
        <div className="hello-world">
            <h1>Hello, World!</h1>
            <p>Welcome to your first React component.</p>
            <ul>
                {items.map((item, index) => (
                    <ListItem key={index} item={item} />
                ))}
            </ul>
        </div>
    );
}

export default HelloWorld;
```

- Используйте ранее созданные компоненты для маршрутов `Counter`, `Form`, и `DataFetcher`.

10. **Заключение**
    - **Рефакторинг кода:**
        - Проверьте, что все компоненты организованы и структурированы.
        - Убедитесь, что нет дублирующегося кода и все компоненты переиспользуются.
    - **Тестирование и отладка:**
        - Проверьте работу всех компонентов и маршрутов.
        - Убедитесь, что приложение работает без ошибок и предупреждений в консоли.
    - **Завершение работы:**
        - Остановите проект, выполнив команду:

          ```
          Ctrl + C
          ```

        - Сохраните все изменения и закройте проект.

Лабораторная работа 2: Управление состоянием в React

Цель лабораторной работы:

Изучить методы управления состоянием компонентов в React с помощью хуков `useState` и `useEffect`, научиться обновлять состояние и реагировать на изменения.

Описание работы:

1. **Использование состояния в компоненте**
   - **Создание компонента `Counter`:**
     - В директории `src` вашего проекта `my-first-react-app` создайте файл `Counter.js`.
     - Добавьте в `Counter.js` следующий код для создания компонента с использованием хука `useState`:

```
import React, { useState } from 'react';

function Counter() {
    const [count, setCount] = useState(0);

    return (
        <div>
            <p>Current count: {count}</p>
            <button onClick={() => setCount(count +
1)}>Increment</button>
            <button onClick={() => setCount(count -
1)}>Decrement</button>
        </div>
    );
}

export default Counter;
```

   - **Использование компонента `Counter` в приложении:**
     - Откройте файл `App.js`.
     - Импортируйте компонент `Counter` и добавьте его в JSX:

```
import React from 'react';
import HelloWorld from './HelloWorld';
import Counter from './Counter';

function App() {
    return (
        <div className="App">
            <HelloWorld />
            <Counter />
        </div>
    );
}

export default App;
```

2. **Использование хука `useEffect`**
   - **Создание компонента `Timer`:**
     - В директории `src` создайте файл `Timer.js`.
     - Добавьте в `Timer.js` следующий код для создания компонента с использованием хука `useEffect`:

```
import React, { useState, useEffect } from 'react';

function Timer() {
    const [seconds, setSeconds] = useState(0);

    useEffect(() => {
        const interval = setInterval(() => {
            setSeconds(seconds => seconds + 1);
        }, 1000);

        return () => clearInterval(interval);
    }, []);

    return (
        <div>
            <p>Elapsed time: {seconds} seconds</p>
        </div>
    );
}

export default Timer;
```

- **Использование компонента `Timer` в приложении:**
  - Откройте файл `App.js`.
  - Импортируйте компонент `Timer` и добавьте его в JSX:

```
import React from 'react';
import HelloWorld from './HelloWorld';
import Counter from './Counter';
import Timer from './Timer';

function App() {
    return (
        <div className="App">
            <HelloWorld />
            <Counter />
            <Timer />
        </div>
    );
}

export default App;
```

3. **Сложное состояние и обработка форм**
   - **Создание компонента для работы с формой:**
     - В директории `src` создайте файл `Form.js`.
     - Добавьте в `Form.js` следующий код для создания компонента с использованием хука `useState` для управления сложным состоянием:

```
import React, { useState } from 'react';

function Form() {
    const [formData, setFormData] = useState({ name: '', email: ''
});

    const handleChange = (e) => {
        const { name, value } = e.target;
        setFormData({ ...formData, [name]: value });
    };

    const handleSubmit = (e) => {
        e.preventDefault();
        alert(`Name: ${formData.name}, Email: ${formData.email}`);
    };

    return (
        <form onSubmit={handleSubmit}>
            <div>
                <label>
                    Name:
                    <input
                        type="text"
                        name="name"
                        value={formData.name}
                        onChange={handleChange}
                    />
                </label>
            </div>
            <div>
                <label>
                    Email:
                    <input
                        type="email"
                        name="email"
                        value={formData.email}
                        onChange={handleChange}
                    />
                </label>
            </div>
            <button type="submit">Submit</button>
        </form>
    );
}

export default Form;
```

- **Использование компонента `Form` в приложении:**
  - Откройте файл `App.js`.
  - Импортируйте компонент `Form` и добавьте его в JSX:

```
import React from 'react';
import HelloWorld from './HelloWorld';
import Counter from './Counter';
import Timer from './Timer';
import Form from './Form';

function App() {
    return (
        <div className="App">
            <HelloWorld />
            <Counter />
            <Timer />
            <Form />
        </div>
    );
}

export default App;
```

4. **Работа с внешними API**
    - **Создание компонента для работы с API:**
        - В директории `src` создайте файл `DataFetcher.js`.
        - Добавьте в `DataFetcher.js` следующий код для создания компонента с использованием хука `useEffect` для получения данных из API:

```
import React, { useState, useEffect } from 'react';

function DataFetcher() {
    const [data, setData] = useState(null);
    const [loading, setLoading] = useState(true);

    useEffect(() => {
        fetch('https://jsonplaceholder.typicode.com/posts/1')
            .then((response) => response.json())
            .then((data) => {
                setData(data);
                setLoading(false);
            });
    }, []);

    if (loading) {
        return <p>Loading...</p>;
    }

    return (
        <div>
            <h3>{data.title}</h3>
            <p>{data.body}</p>
        </div>
    );
}

export default DataFetcher;
```

- **Использование компонента `DataFetcher` в приложении:**
  - Откройте файл `App.js`.
  - Импортируйте компонент `DataFetcher` и добавьте его в JSX:

```
import React from 'react';
import HelloWorld from './HelloWorld';
import Counter from './Counter';
import Timer from './Timer';
import Form from './Form';
import DataFetcher from './DataFetcher';

function App() {
    return (
        <div className="App">
            <HelloWorld />
            <Counter />
            <Timer />
            <Form />
            <DataFetcher />
        </div>
    );
}

export default App;
```

5. **Заключение**
   - **Рефакторинг кода:**
     - Проверьте, что все компоненты организованы и структурированы.
     - Убедитесь, что нет дублирующегося кода и все компоненты переиспользуются.
   - **Тестирование и отладка:**
     - Проверьте работу всех компонентов и их взаимодействие.
     - Убедитесь, что приложение работает без ошибок и предупреждений в консоли.
   - **Завершение работы:**
     - Остановите проект, выполнив команду:

       ```
       Ctrl + C
       ```

     - Сохраните все изменения и закройте проект.

Лабораторная работа 3: Роутинг и навигация в React

Цель лабораторной работы:

Познакомиться с библиотекой React Router и научиться создавать маршруты для навигации между страницами в веб-приложении на React.

Описание работы:

1. **Установка и настройка React Router**
   - **Установка библиотеки React Router:**
     - В командной строке выполните команду для установки `react-router-dom`:

```
npm install react-router-dom
```

   - **Настройка маршрутов в приложении:**
     - Откройте файл `App.js` и добавьте маршрутизацию:

```
import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-
dom';
import Navbar from './Navbar';
import Home from './Home';
import About from './About';
import Contact from './Contact';
import Post from './Post';

function App() {
    return (
        <Router>
            <div className="App">
                <Navbar />
                <Switch>
                    <Route path="/" exact component={Home} />
                    <Route path="/about" component={About} />
                    <Route path="/contact" component={Contact} />
                    <Route path="/post/:id" component={Post} />
                </Switch>
            </div>
        </Router>
    );
}

export default App;
```

2. **Создание компонентов для страниц**
   - **Home.js**

```
import React from 'react';

function Home() {
    return (
        <div>
            <h1>Home Page</h1>
            <p>Welcome to the home page of our application.</p>
        </div>
    );
}

export default Home;
```

- **About.js:**

```
import React from 'react';

function About() {
    return (
        <div>
            <h1>About Page</h1>
            <p>This is the about page of our application.</p>
        </div>
    );
}

export default About;
```

- **Contact.js:**

```
import React from 'react';

function Contact() {
    return (
        <div>
            <h1>Contact Page</h1>
            <p>This is the contact page of our application.</p>
        </div>
    );
}

export default Contact;
```

- **Post.js:**

```
import React, { useEffect, useState } from 'react';
import { useParams } from 'react-router-dom';
function Post() {
    const { id } = useParams();
    const [post, setPost] = useState(null);

    useEffect(() => {
        fetch(`https://jsonplaceholder.typicode.com/posts/${id}`)
            .then(response => response.json())
            .then(data => setPost(data));
    }, [id]);
    if (!post) {
        return <p>Loading...</p>;
    }

    return (
        <div>
            <h1>{post.title}</h1>
            <p>{post.body}</p>
        </div>
    );
}

export default Post;
```

3. **Создание навигационного меню**
    - **Navbar.js:**

```
import React from 'react';
import { Link } from 'react-router-dom';

function Navbar() {
    return (
        <nav>
            <ul>
                <li><Link to="/">Home</Link></li>
                <li><Link to="/about">About</Link></li>
                <li><Link to="/contact">Contact</Link></li>
            </ul>
        </nav>
    );
}

export default Navbar;
```

4. **Добавление динамического роутинга для постов**
    - **Добавление списка постов и ссылок на отдельные посты:**
        - Создайте файл `Posts.js`:

```
import React, { useEffect, useState } from 'react';
import { Link } from 'react-router-dom';

function Posts() {
    const [posts, setPosts] = useState([]);

    useEffect(() => {
        fetch('https://jsonplaceholder.typicode.com/posts')
            .then(response => response.json())
            .then(data => setPosts(data));
    }, []);

    return (
        <div>
            <h1>Posts</h1>
            <ul>
                {posts.map(post => (
                    <li key={post.id}>
                        <Link
to={`/post/${post.id}`}>{post.title}</Link>
                    </li>
                ))}
            </ul>
        </div>
    );
}

export default Posts;
```

        - Обновите файл `App.js` для добавления маршрута к списку постов:

```
import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-
dom';
import Navbar from './Navbar';
import Home from './Home';
import About from './About';
import Contact from './Contact';
import Posts from './Posts';
import Post from './Post';

function App() {
    return (
        <Router>
            <div className="App">
                <Navbar />
                <Switch>
                    <Route path="/" exact component={Home} />
                    <Route path="/about" component={About} />
                    <Route path="/contact" component={Contact} />
                    <Route path="/posts" component={Posts} />
                    <Route path="/post/:id" component={Post} />
                </Switch>
            </div>
        </Router>
    );
}

export default App;
```

- Обновите файл `Navbar.js` для добавления ссылки на страницу постов:

```
import React from 'react';
import { Link } from 'react-router-dom';

function Navbar() {
    return (
        <nav>
            <ul>
                <li><Link to="/">Home</Link></li>
                <li><Link to="/about">About</Link></li>
                <li><Link to="/contact">Contact</Link></li>
                <li><Link to="/posts">Posts</Link></li>
            </ul>
        </nav>
    );
}

export default Navbar;
```

5. **Заключение**
    - **Рефакторинг кода:**
        - Проверьте, что все компоненты организованы и структурированы.
        - Убедитесь, что нет дублирующегося кода и все компоненты переиспользуются.
    - **Тестирование и отладка:**
        - Проверьте работу всех компонентов и их взаимодействие.

- Убедитесь, что приложение работает без ошибок и предупреждений в консоли.

- **Завершение работы:**
  - Сохраните все изменения и закройте проект.
  -

Лабораторная работа 4: Стилизация компонентов в React

Цель лабораторной работы:

Изучить различные способы стилизации компонентов в React, включая CSS Modules и Styled Components, научиться создавать стилизованные интерфейсы.

Описание работы:

1. **Использование CSS Modules**
   - **Создание компонента с использованием CSS Modules:**
     - В директории `src` создайте файл `Button.module.css`.
     - Добавьте в `Button.module.css` следующий код:

```
.button {
    background-color: #4CAF50;
    border: none;
    color: white;
    padding: 15px 32px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
    margin: 4px 2px;
    cursor: pointer;
    border-radius: 4px;
}
```

     - В директории `src` создайте файл `Button.js`.
     - Добавьте в `Button.js` следующий код для создания компонента с использованием CSS Modules:

```
import React from 'react';
import styles from './Button.module.css';

function Button({ label }) {
    return <button className={styles.button}>{label}</button>;
}

export default Button;
```

   - **Использование компонента `Button` в приложении:**
     - Откройте файл `App.js`.
     - Импортируйте компонент `Button` и добавьте его в JSX:

```
import React from 'react';
import Navbar from './Navbar';
import Home from './Home';
import About from './About';
import Contact from './Contact';
import HelloWorld from './HelloWorld';
import Counter from './Counter';
import Timer from './Timer';
import Form from './Form';
import DataFetcher from './DataFetcher';
import Post from './Post';
import Button from './Button';

function App() {
    return (
        <Router>
            <div className="App">
                <Navbar />
                <Switch>
                    <Route path="/" exact component={Home} />
                    <Route path="/about" component={About} />
                    <Route path="/contact" component={Contact} />
                    <Route path="/helloworld" component={HelloWorld}
/>
                    <Route path="/counter" component={Counter} />
                    <Route path="/timer" component={Timer} />
                    <Route path="/form" component={Form} />
                    <Route path="/data-fetcher"
component={DataFetcher} />
                    <Route path="/post/:id" component={Post} />
                </Switch>
                <Button label="Click Me" />
            </div>
        </Router>
    );
}

export default App;
```

2. **Использование Styled Components**
   - **Установка библиотеки:**
     - В командной строке выполните команду для установки `styled-components`:

```
npm install styled-components
```

   - **Создание компонента с использованием Styled Components:**
     - В директории `src` создайте файл `StyledButton.js`.
     - Добавьте в `StyledButton.js` следующий код для создания компонента с использованием Styled Components:

```
import React from 'react';
import styled from 'styled-components';

const Button = styled.button`
    background-color: #4CAF50;
    border: none;
    color: white;
    padding: 15px 32px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
    margin: 4px 2px;
    cursor: pointer;
    border-radius: 4px;
`;

function StyledButton({ label }) {
    return <Button>{label}</Button>;
}

export default StyledButton;
```

- **Использование компонента `StyledButton` в приложении:**
  - Откройте файл `App.js`.
  - Импортируйте компонент `StyledButton` и добавьте его в JSX:

```
import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-
dom';
import Navbar from './Navbar';
import Home from './Home';
import About from './About';
import Contact from './Contact';
import HelloWorld from './HelloWorld';
import Counter from './Counter';
import Timer from './Timer';
import Form from './Form';
import DataFetcher from './DataFetcher';
import Post from './Post';
import Button from './Button';
import StyledButton from './StyledButton';

function App() {
    return (
        <Router>
            <div className="App">
                <Navbar />
                <Switch>
                    <Route path="/" exact component={Home} />
                    <Route path="/about" component={About} />
                    <Route path="/contact" component={Contact} />
                    <Route path="/helloworld" component={HelloWorld}
/>
                    <Route path="/counter" component={Counter} />
                    <Route path="/timer" component={Timer} />
                    <Route path="/form" component={Form} />
                    <Route path="/data-fetcher"
component={DataFetcher} />
                    <Route path="/post/:id" component={Post} />
                </Switch>
                <Button label="Click Me" />
                <StyledButton label="Styled Click Me" />
            </div>
        </Router>
    );
}

export default App;
```

3. **Стилизация с помощью inline-стилей**
    - **Создание компонента с inline-стилями:**
        - В директории `src` создайте файл `InlineStyledComponent.js`.
        - Добавьте в `InlineStyledComponent.js` следующий код для создания компонента с использованием inline-стилей:

```
import React from 'react';

function InlineStyledComponent() {
    const style = {
        backgroundColor: '#4CAF50',
        color: 'white',
        padding: '15px 32px',
        textAlign: 'center',
        textDecoration: 'none',
        display: 'inline-block',
        fontSize: '16px',
        margin: '4px 2px',
        cursor: 'pointer',
        borderRadius: '4px',
    };

    return <button style={style}>Inline Styled Button</button>;
}

export default InlineStyledComponent;
```

- **Использование компонента `InlineStyledComponent` в приложении:**
  - Откройте файл `App.js`.
  - Импортируйте компонент `InlineStyledComponent` и добавьте его в JSX:

```
import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-
dom';
import Navbar from './Navbar';
import Home from './Home';
import About from './About';
import Contact from './Contact';
import HelloWorld from './HelloWorld';
import Counter from './Counter';
import Timer from './Timer';
import Form from './Form';
import DataFetcher from './DataFetcher';
import Post from './Post';
import Button from './Button';
import StyledButton from './StyledButton';
import InlineStyledComponent from './InlineStyledComponent';

function App() {
    return (
        <Router>
            <div className="App">
                <Navbar />
                <Switch>
                    <Route path="/" exact component={Home} />
                    <Route path="/about" component={About} />
                    <Route path="/contact" component={Contact} />
                    <Route path="/helloworld" component={HelloWorld}
/>
                    <Route path="/counter" component={Counter} />
                    <Route path="/timer" component={Timer} />
                    <Route path="/form" component={Form} />
                    <Route path="/data-fetcher"
component={DataFetcher} />
                    <Route path="/post/:id" component={Post} />
                </Switch>
                <Button label="Click Me" />
                <StyledButton label="Styled Click Me" />
                <InlineStyledComponent />
            </div>
        </Router>
    );
}

export default App;
```

4. **Использование глобальных стилей**
    - **Создание глобальных стилей:**
        - В директории `src` создайте файл `global.css`.
        - Добавьте в `global.css` следующий код для создания глобальных стилей:

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

.App {
    text-align: center;
}
```

- **Импорт глобальных стилей в `index.js`:**
  - Обновите файл `index.js` для импорта глобальных стилей:

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import './global.css';
import App from './App';

ReactDOM.render(
    <React.StrictMode>
        <App />
    </React.StrictMode>,
    document.getElementById('root')
);
```

5. **Заключение**
   - **Рефакторинг кода:**
     - Проверьте, что все стили организованы и структурированы.
     - Убедитесь, что нет дублирующихся стилей и все компоненты переиспользуются.

Лабораторная работа 5: Формы и обработка событий в React

Цель лабораторной работы:

Практически изучить создание и валидацию форм в React, научиться обрабатывать события в компонентах для реализации интерактивности.

Описание работы:

1. **Создание формы в React**
   - **Создание компонента формы:**
     - В директории `src` создайте файл `ContactForm.js`.
     - Добавьте в `ContactForm.js` следующий код для создания формы:

```
import React, { useState } from 'react';

function ContactForm() {
    const [formData, setFormData] = useState({ name: '', email: '',
message: '' });
    const [errors, setErrors] = useState({});

    const handleChange = (e) => {
        const { name, value } = e.target;
        setFormData({ ...formData, [name]: value });
    };

    const validate = () => {
        let inputErrors = {};
        if (!formData.name) inputErrors.name = 'Name is required';
        if (!formData.email) {
            inputErrors.email = 'Email is required';
        } else if (!/\S+@\S+\.\S+/.test(formData.email)) {
            inputErrors.email = 'Email is invalid';
        }
        if (!formData.message) inputErrors.message = 'Message is
required';
        return inputErrors;
    };

    const handleSubmit = (e) => {
        e.preventDefault();
        const validationErrors = validate();
        if (Object.keys(validationErrors).length > 0) {
            setErrors(validationErrors);
        } else {
            alert(`Name: ${formData.name}, Email: ${formData.email},
Message: ${formData.message}`);
            setFormData({ name: '', email: '', message: '' });
            setErrors({});
        }
    };

    return (
        <form onSubmit={handleSubmit}>
            <div>
                <label>
                    Name:
                    <input
                        type="text"
                        name="name"
                        value={formData.name}
                        onChange={handleChange}
                    />
                </label>
                {errors.name && <p style={{ color: 'red'
}}>{errors.name}</p>}
            </div>
            <div>
                <label>
                    Email:
                    <input
                        type="email"
                        name="email"
                        value={formData.email}
                        onChange={handleChange}
                    />
```

```
                </label>
                {errors.email && <p style={{ color: 'red'
}}>{errors.email}</p>}
            </div>
            <div>
                <label>
                    Message:
                    <textarea
                        name="message"
                        value={formData.message}
                        onChange={handleChange}
                    />
                </label>
                {errors.message && <p style={{ color: 'red'
}}>{errors.message}</p>}
            </div>
            <button type="submit">Submit</button>
        </form>
    );
}
export default ContactForm;
```

- **Использование компонента `ContactForm` в приложении:**
  - Откройте файл `App.js`.
  - Импортируйте компонент `ContactForm` и добавьте его в JSX:

```
import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-
dom';
import Header from './components/Header';
import Footer from './components/Footer';
import Home from './pages/Home';
import About from './pages/About';
import Contact from './pages/Contact';
import Posts from './pages/Posts';
import Post from './pages/Post';
import ContactForm from './ContactForm';
import Navbar from './components/Navbar';
function App() {
    return (
        <Router>
            <div className="App">
                <Header />
                <Navbar />
                <Switch>
                    <Route path="/" exact component={Home} />
                    <Route path="/about" component={About} />
                    <Route path="/contact" component={Contact} />
                    <Route path="/posts" component={Posts} />
                    <Route path="/post/:id" component={Post} />
                </Switch>
                <ContactForm />
                <Footer />
            </div>
        </Router>
    );
}

export default App;
```

2. **Обработка событий в компонентах**
    - **Создание компонента с обработкой событий:**
        - В директории `src` создайте файл `EventComponent.js`.
        - Добавьте в `EventComponent.js` следующий код для создания компонента, обрабатывающего события:

```
import React, { useState } from 'react';

function EventComponent() {
    const [isHovered, setIsHovered] = useState(false);

    const handleMouseEnter = () => {
        setIsHovered(true);
    };

    const handleMouseLeave = () => {
        setIsHovered(false);
    };

    return (
        <div
            onMouseEnter={handleMouseEnter}
            onMouseLeave={handleMouseLeave}
            style={{
                backgroundColor: isHovered ? 'lightblue' :
'lightgray',
                padding: '20px',
                borderRadius: '8px',
                textAlign: 'center',
            }}
        >
            {isHovered ? 'You are hovering over me!' : 'Hover over
me!'}
        </div>
    );
}

export default EventComponent;
```

    - **Использование компонента `EventComponent` в приложении:**
        - Откройте файл `App.js`.
        - Импортируйте компонент `EventComponent` и добавьте его в JSX:

```
import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-
dom';
import Header from './components/Header';
import Footer from './components/Footer';
import Home from './pages/Home';
import About from './pages/About';
import Contact from './pages/Contact';
import Posts from './pages/Posts';
import Post from './pages/Post';
import ContactForm from './ContactForm';
import EventComponent from './EventComponent';
import Navbar from './components/Navbar';

function App() {
    return (
        <Router>
            <div className="App">
                <Header />
                <Navbar />
                <Switch>
                    <Route path="/" exact component={Home} />
                    <Route path="/about" component={About} />
                    <Route path="/contact" component={Contact} />
                    <Route path="/posts" component={Posts} />
                    <Route path="/post/:id" component={Post} />
                </Switch>
                <ContactForm />
                <EventComponent />
                <Footer />
            </div>
        </Router>
    );
}

export default App;
```

3. **Валидация форм**
    - **Добавление валидации в компонент `ContactForm`:**
        - Обновите компонент `ContactForm` для включения валидации:

```
import React, { useState } from 'react';

function ContactForm() {
    const [formData, setFormData] = useState({ name: '', email: '',
message: '' });
    const [errors, setErrors] = useState({});

    const handleChange = (e) => {
        const { name, value } = e.target;
        setFormData({ ...formData, [name]: value });
    };

    const validate = () => {
        let inputErrors = {};
        if (!formData.name) inputErrors.name = 'Name is required';
        if (!formData.email) {
            inputErrors.email = 'Email is required';
        } else if (!/\S+@\S+\.\S+/.test(formData.email)) {
            inputErrors.email = 'Email is invalid';
```

```
        }
        if (!formData.message) inputErrors.message = 'Message is
required';
        return inputErrors;
    };

    const handleSubmit = (e) => {
        e.preventDefault();
        const validationErrors = validate();
        if (Object.keys(validationErrors).length > 0) {
            setErrors(validationErrors);
        } else {
            alert(`Name: ${formData.name}, Email: ${formData.email},
Message: ${formData.message}`);
            setFormData({ name: '', email: '', message: '' });
            setErrors({});
        }
    };

    return (
        <form onSubmit={handleSubmit}>
            <div>
                <label>
                    Name:
                    <input
                        type="text"
                        name="name"
                        value={formData.name}
                        onChange={handleChange}
                    />
                </label>
                {errors.name && <p style={{ color: 'red'
}}>{errors.name}</p>}
            </div>
            <div>
                <label>
                    Email:
                    <input
                        type="email"
                        name="email"
                        value={formData.email}
                        onChange={handleChange}
                    />
                </label>
                {errors.email && <p style={{ color: 'red'
}}>{errors.email}</p>}
            </div>
            <div>
                <label>
                    Message:
                    <textarea
                        name="message"
                        value={formData.message}
                        onChange={handleChange}
                    />
                </label>
                {errors.message && <p style={{ color: 'red'
}}>{errors.message}</p>}
            </div>
            <button type="submit">Submit</button>
        </form>
    );
```

```
}

export default ContactForm;
```

4. **Создание более сложной формы**
   - **Создание компонента для сложной формы с дополнительными полями:**
     - В директории `src` создайте файл `ComplexForm.js`.
     - Добавьте в `ComplexForm.js` следующий код для создания сложной формы:

```
import React, { useState } from 'react';

function ComplexForm() {
    const [formData, setFormData] = useState({ name: '', email: '',
message: '', age: '', gender: '' });
    const [errors, setErrors] = useState({});

    const handleChange = (e) => {
        const { name, value } = e.target;
        setFormData({ ...formData, [name]: value });
    };

    const validate = () => {
        let inputErrors = {};
        if (!formData.name) inputErrors.name = 'Name is required';
        if (!formData.email) {
            inputErrors.email = 'Email is required';
        } else if (!/\S+@\S+\.\S+/.test(formData.email)) {
            inputErrors.email = 'Email is invalid';
        }
        if (!formData.message) inputErrors.message = 'Message is
required';
        if (!formData.age) inputErrors.age = 'Age is required';
        if (!formData.gender) inputErrors.gender = 'Gender is
required';
        return inputErrors;
    };

    const handleSubmit = (e) => {
        e.preventDefault();
        const validationErrors = validate();
        if (Object.keys(validationErrors).length > 0) {
            setErrors(validationErrors);
        } else {
            alert(`Form submitted with data:
${JSON.stringify(formData)}`);
            setFormData({ name: '', email: '', message: '', age: '',
gender: '' });
            setErrors({});
        }
    };

    return (
        <form onSubmit={handleSubmit}>
            <div>
                <label>
                    Name:
                    <input
                        type="text"
```

```
                        name="name"
                        value={formData.name}
                        onChange={handleChange}
                    />
                </label>
                {errors.name && <p style={{ color: 'red'
}}>{errors.name}</p>}
            </div>
            <div>
                <label>
                    Email:
                    <input
                        type="email"
                        name="email"
                        value={formData.email}
                        onChange={handleChange}
                    />
                </label>
                {errors.email && <p style={{ color: 'red'
}}>{errors.email}</p>}
            </div>
            <div>
                <label>
                    Message:
                    <textarea
                        name="message"
                        value={formData.message}
                        onChange={handleChange}
                    />
                </label>
                {errors.message && <p style={{ color: 'red'
}}>{errors.message}</p>}
            </div>
            <div>
                <label>
                    Age:
                    <input
                        type="number"
                        name="age"
                        value={formData.age}
                        onChange={handleChange}
                    />
                </label>
                {errors.age && <p style={{ color: 'red'
}}>{errors.age}</p>}
            </div>
            <div>
                <label>
                    Gender:
                    <select name="gender" value={formData.gender}
onChange={handleChange}>
                        <option value="">Select...</option>
                        <option value="male">Male</option>
                        <option value="female">Female</option>
                        <option value="other">Other</option>
                    </select>
                </label>
                {errors.gender && <p style={{ color: 'red'
}}>{errors.gender}</p>}
            </div>
            <button type="submit">Submit</button>
        </form>
```

```
    );
}

export default ComplexForm;
```

- **Использование компонента `ComplexForm` в приложении:**
  - Откройте файл `App.js`.
  - Импортируйте компонент `ComplexForm` и добавьте его в JSX:

```
import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-
dom';
import Header from './components/Header';
import Footer from './components/Footer';
import Home from './pages/Home';
import About from './pages/About';
import Contact from './pages/Contact';
import Posts from './pages/Posts';
import Post from './pages/Post';
import ContactForm from './ContactForm';
import EventComponent from './EventComponent';
import ComplexForm from './ComplexForm';
import Navbar from './components/Navbar';

function App() {
    return (
        <Router>
            <div className="App">
                <Header />
                <Navbar />
                <Switch>
                    <Route path="/" exact component={Home} />
                    <Route path="/about" component={About} />
                    <Route path="/contact" component={Contact} />
                    <Route path="/posts" component={Posts} />
                    <Route path="/post/:id" component={Post} />
                </Switch>
                <ContactForm />
                <EventComponent />
                <ComplexForm />
                <Footer />
            </div>
        </Router>
    );
}

export default App;
```

5. **Заключение**
   - **Рефакторинг кода:**
     - Проверьте, что все компоненты организованы и структурированы.
     - Убедитесь, что нет дублирующегося кода и все компоненты переиспользуются.
   - **Тестирование и отладка:**
     - Проверьте работу всех компонентов и их взаимодействие.

- Убедитесь, что приложение работает без ошибок и предупреждений в консоли.

- **Завершение работы:**
  - Сохраните все изменения и закройте проект.

Лабораторная работа 6: Взаимодействие с сервером в React

Цель лабораторной работы:

Освоить основы взаимодействия с сервером в React-приложениях с использованием Fetch API для отправки запросов на сервер и получения данных.

Описание работы:

1. **Настройка сервера для взаимодействия**
   - **Создание простого сервера с использованием JSON Server:**
     - Установите JSON Server глобально:

```
npm install -g json-server
```

     - В корне вашего проекта создайте файл `db.json` со следующими данными:

```
json

{
    "posts": [
        { "id": 1, "title": "Post 1", "body": "This is the body of
post 1." },
        { "id": 2, "title": "Post 2", "body": "This is the body of
post 2." },
        { "id": 3, "title": "Post 3", "body": "This is the body of
post 3." }
    ]
}
```

     - Запустите JSON Server:

```
json-server --watch db.json --port 5000
```

2. **Получение данных с сервера**
   - **Создание компонента для получения данных:**
     - В директории `src` создайте файл `PostList.js`.
     - Добавьте в `PostList.js` следующий код для получения данных с сервера:

```
import React, { useState, useEffect } from 'react';

function PostList() {
    const [posts, setPosts] = useState([]);
    const [loading, setLoading] = useState(true);

    useEffect(() => {
        fetch('http://localhost:5000/posts')
            .then(response => response.json())
            .then(data => {
                setPosts(data);
                setLoading(false);
            });
    }, []);

    if (loading) {
        return <p>Loading...</p>;
    }

    return (
        <div>
            <h1>Posts</h1>
            <ul>
                {posts.map(post => (
                    <li key={post.id}>
                        <h2>{post.title}</h2>
                        <p>{post.body}</p>
                    </li>
                ))}
            </ul>
        </div>
    );
}

export default PostList;
```

- **Использование компонента `PostList` в приложении:**
  - Откройте файл `App.js`.
  - Импортируйте компонент `PostList` и добавьте его в JSX:

```
import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-
dom';
import Navbar from './Navbar';
import Home from './Home';
import About from './About';
import Contact from './Contact';
import HelloWorld from './HelloWorld';
import Counter from './Counter';
import Timer from './Timer';
import Form from './Form';
import DataFetcher from './DataFetcher';
import Post from './Post';
import Button from './Button';
import StyledButton from './StyledButton';
import InlineStyledComponent from './InlineStyledComponent';
import ContactForm from './ContactForm';
import EventComponent from './EventComponent';
import PostList from './PostList';
```

```
function App() {
    return (
        <Router>
            <div className="App">
                <Navbar />
                <Switch>
                    <Route path="/" exact component={Home} />
                    <Route path="/about" component={About} />
                    <Route path="/contact" component={Contact} />
                    <Route path="/helloworld" component={HelloWorld}
/>
                    <Route path="/counter" component={Counter} />
                    <Route path="/timer" component={Timer} />
                    <Route path="/form" component={Form} />
                    <Route path="/data-fetcher"
component={DataFetcher} />
                    <Route path="/post/:id" component={Post} />
                    <Route path="/posts" component={PostList} />
                </Switch>
                <Button label="Click Me" />
                <StyledButton label="Styled Click Me" />
                <InlineStyledComponent />
                <ContactForm />
                <EventComponent />
            </div>
        </Router>
    );
}

export default App;
```

3. **Отправка данных на сервер**
    - **Создание компонента для отправки данных:**
        - В директории `src` создайте файл `CreatePost.js`.
        - Добавьте в `CreatePost.js` следующий код для создания формы и отправки данных на сервер:

```
import React, { useState } from 'react';

function CreatePost() {
    const [title, setTitle] = useState('');
    const [body, setBody] = useState('');

    const handleSubmit = (e) => {
        e.preventDefault();
        const newPost = { title, body };

        fetch('http://localhost:5000/posts', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify(newPost),
        })
            .then(response => response.json())
            .then(data => {
                alert('Post created!');
                setTitle('');
```

```
                setBody('');
            });
    };

    return (
        <form onSubmit={handleSubmit}>
            <div>
                <label>
                    Title:
                    <input
                        type="text"
                        value={title}
                        onChange={(e) => setTitle(e.target.value)}
                    />
                </label>
            </div>
            <div>
                <label>
                    Body:
                    <textarea
                        value={body}
                        onChange={(e) => setBody(e.target.value)}
                    />
                </label>
            </div>
            <button type="submit">Create Post</button>
        </form>
    );
}

export default CreatePost;
```

- **Использование компонента `CreatePost` в приложении:**
  - Откройте файл `App.js`.
  - Импортируйте компонент `CreatePost` и добавьте его в JSX:

```
import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-
dom';
import Navbar from './Navbar';
import Home from './Home';
import About from './About';
import Contact from './Contact';
import HelloWorld from './HelloWorld';
import Counter from './Counter';
import Timer from './Timer';
import Form from './Form';
import DataFetcher from './DataFetcher';
import Post from './Post';
import Button from './Button';
import StyledButton from './StyledButton';
import InlineStyledComponent from './InlineStyledComponent';
import ContactForm from './ContactForm';
import EventComponent from './EventComponent';
import PostList from './PostList';
import CreatePost from './CreatePost';

function App() {
    return (
        <Router>
```

```
            <div className="App">
                <Navbar />
                <Switch>
                    <Route path="/" exact component={Home} />
                    <Route path="/about" component={About} />
                    <Route path="/contact" component={Contact} />
                    <Route path="/helloworld" component={HelloWorld}
/>
                    <Route path="/counter" component={Counter} />
                    <Route path="/timer" component={Timer} />
                    <Route path="/form" component={Form} />
                    <Route path="/data-fetcher"
component={DataFetcher} />
                    <Route path="/post/:id" component={Post} />
                    <Route path="/posts" component={PostList} />
                    <Route path="/create-post" component={CreatePost}
/>
                </Switch>
                <Button label="Click Me" />
                <StyledButton label="Styled Click Me" />
                <InlineStyledComponent />
                <ContactForm />
                <EventComponent />
            </div>
        </Router>
    );
}

export default App;
```

4. **Обновление и удаление данных**
    - **Создание компонентов для обновления и удаления данных:**
        - Обновите файл `PostList.js` для добавления функциональности обновления и удаления постов:

```
import React, { useState, useEffect } from 'react';

function PostList() {
    const [posts, setPosts] = useState([]);
    const [loading, setLoading] = useState(true);

    useEffect(() => {
        fetch('http://localhost:5000/posts')
            .then(response => response.json())
            .then(data => {
                setPosts(data);
                setLoading(false);
            });
    }, []);

    const handleDelete = (id) => {
        fetch(`http://localhost:5000/posts/${id}`, {
            method: 'DELETE',
        }).then(() => {
            setPosts(posts.filter(post => post.id !== id));
        });
    };

    const handleUpdate = (id) => {
```

```
        const updatedTitle = prompt('Enter new title:');
        const updatedBody = prompt('Enter new body:');
        if (updatedTitle && updatedBody) {
            const updatedPost = { title: updatedTitle, body:
updatedBody };

            fetch(`http://localhost:5000/posts/${id}`, {
                method: 'PUT',
                headers: {
                    'Content-Type': 'application/json',
                },
                body: JSON.stringify(updatedPost),
            })
                .then(response => response.json())
                .then(data => {
                    setPosts(posts.map(post => (post.id ===
setPosts(posts.map(post => (post.id === id ? data : post)));
                });
            }
        };

        if (loading) {
            return <p>Loading...</p>;
        }

        return (
            <div>
                <h1>Posts</h1>
                <ul>
                    {posts.map(post => (
                        <li key={post.id}>
                            <h2>{post.title}</h2>
                            <p>{post.body}</p>
                            <button onClick={() =>
handleDelete(post.id)}>Delete</button>
                            <button onClick={() =>
handleUpdate(post.id)}>Update</button>
                        </li>
                    ))}
                </ul>
            </div>
        );
    }

    export default PostList;
```

5. **Использование компонентов `PostList` и `CreatePost` в приложении**
   - Откройте файл `App.js`.
   - Импортируйте компоненты `PostList` и `CreatePost` и добавьте их в JSX:

```
import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';
import Navbar from './Navbar';
import Home from './Home';
import About from './About';
import Contact from './Contact';
import HelloWorld from './HelloWorld';
import Counter from './Counter';
import Timer from './Timer';
```

```
import Form from './Form';
import DataFetcher from './DataFetcher';
import Post from './Post';
import Button from './Button';
import StyledButton from './StyledButton';
import InlineStyledComponent from './InlineStyledComponent';
import ContactForm from './ContactForm';
import EventComponent from './EventComponent';
import PostList from './PostList';
import CreatePost from './CreatePost';

function App() {
    return (
        <Router>
            <div className="App">
                <Navbar />
                <Switch>
                    <Route path="/" exact component={Home} />
                    <Route path="/about" component={About} />
                    <Route path="/contact" component={Contact} />
                    <Route path="/helloworld" component={HelloWorld} />
                    <Route path="/counter" component={Counter} />
                    <Route path="/timer" component={Timer} />
                    <Route path="/form" component={Form} />
                    <Route path="/data-fetcher" component={DataFetcher} />
                    <Route path="/post/:id" component={Post} />
                    <Route path="/posts" component={PostList} />
                    <Route path="/create-post" component={CreatePost} />
                </Switch>
                <Button label="Click Me" />
                <StyledButton label="Styled Click Me" />
                <InlineStyledComponent />
                <ContactForm />
                <EventComponent />
            </div>
        </Router>
    );
}

export default App;
```

6. **Заключение**
    - **Рефакторинг кода:**
        - Проверьте, что все компоненты организованы и структурированы.
        - Убедитесь, что нет дублирующегося кода и все компоненты переиспользуются.
    - **Тестирование и отладка:**
        - Проверьте работу всех компонентов и их взаимодействие.
        - Убедитесь, что приложение работает без ошибок и предупреждений в консоли.
    - **Завершение работы:**
        - Остановите проект, выполнив команду:

          ```
          Ctrl + C
          ```

        - Сохраните все изменения и закройте проект.

Лабораторная работа 7: Оптимизация производительности в React

Цель лабораторной работы:

Изучить методы оптимизации производительности React-приложений, включая мемоизацию, `useMemo` и `useCallback`, и применить их на практике.

Описание работы:

1. **Использование мемоизации с `React.memo`**
   - **Создание компонента для мемоизации:**
     - В директории `src` создайте файл `MemoizedComponent.js`.
     - Добавьте в `MemoizedComponent.js` следующий код для создания мемоизированного компонента:

```
import React from 'react';

function MemoizedComponent({ value }) {
    console.log('Rendering MemoizedComponent');
    return <div>Memoized Value: {value}</div>;
}

export default React.memo(MemoizedComponent);
```

2. **Использование `useMemo` для мемоизации вычислений**
   - **Создание компонента с `useMemo`:**
     - В директории `src` создайте файл `ExpensiveCalculation.js`.
     - Добавьте в `ExpensiveCalculation.js` следующий код:

```
import React, { useState, useMemo } from 'react';

function ExpensiveCalculation() {
    const [count, setCount] = useState(0);
    const [value, setValue] = useState('');

    const expensiveCalculation = useMemo(() => {
        console.log('Calculating...');
        let result = 0;
        for (let i = 0; i < 1000000000; i++) {
            result += i;
        }
        return result;
    }, [count]);

    return (
        <div>
            <h1>Expensive Calculation</h1>
            <div>Result: {expensiveCalculation}</div>
```

```
            <button onClick={() => setCount(count + 1)}>Increment
Count</button>
            <input
                type="text"
                value={value}
                onChange={(e) => setValue(e.target.value)}
                placeholder="Type something..."
            />
        </div>
    );
}

export default ExpensiveCalculation;
```

3. **Использование `useCallback` для мемоизации функций**
   - **Создание компонента с `useCallback`:**
     - В директории `src` создайте файл `CallbackComponent.js`.
     - Добавьте в `CallbackComponent.js` следующий код:

```
import React, { useState, useCallback } from 'react';

function CallbackComponent() {
    const [count, setCount] = useState(0);
    const [value, setValue] = useState('');

    const handleButtonClick = useCallback(() => {
        console.log('Button clicked!');
    }, []);

    return (
        <div>
            <h1>Callback Component</h1>
            <button onClick={() => setCount(count + 1)}>Increment
Count</button>
            <button onClick={handleButtonClick}>Click Me</button>
            <input
                type="text"
                value={value}
                onChange={(e) => setValue(e.target.value)}
                placeholder="Type something..."
            />
        </div>
    );
}

export default CallbackComponent;
```

4. **Использование `React.Profiler` для измерения производительности**
   - **Создание компонента с использованием `React.Profiler`:**
     - В директории `src` создайте файл `ProfilerComponent.js`.
     - Добавьте в `ProfilerComponent.js` следующий код:

```
import React, { Profiler } from 'react';

function ProfilerComponent() {
```

```
    const callback = (
        id, // the "id" prop of the Profiler tree that has just
committed
        phase, // either "mount" (if the tree just mounted) or
"update" (if it re-rendered)
        actualDuration, // time spent rendering the committed update
        baseDuration, // estimated time to render the entire subtree
without memoization
        startTime, // when React began rendering this update
        commitTime, // when React committed this update
        interactions // the Set of interactions belonging to this
update
    ) => {
        console.log({ id, phase, actualDuration, baseDuration,
startTime, commitTime, interactions });
    };

    return (
        <Profiler id="profiler" onRender={callback}>
            <div>
                <h1>Profiler Component</h1>
                <p>This component uses React Profiler to measure
performance.</p>
            </div>
        </Profiler>
    );
}

export default ProfilerComponent;
```

5. **Полный файл App.js с включенными всеми компонентами**
   - App.js:

```
import React, { useState } from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';
import Home from './Home';
import About from './About';
import Contact from './Contact';
import Navbar from './Navbar';
import MemoizedComponent from './MemoizedComponent';
import ExpensiveCalculation from './ExpensiveCalculation';
import CallbackComponent from './CallbackComponent';
import ProfilerComponent from './ProfilerComponent';

function App() {
    const [count, setCount] = useState(0);

    return (
        <Router>
            <div className="App">
                <Navbar />
                <Switch>
                    <Route path="/" exact component={Home} />
                    <Route path="/about" component={About} />
                    <Route path="/contact" component={Contact} />
                </Switch>
                <button onClick={() => setCount(count + 1)}>Increment
Count</button>
                <MemoizedComponent value={count} />
                <ExpensiveCalculation />
                <CallbackComponent />
```

```
                <ProfilerComponent />
            </div>
        </Router>
    );
}

export default App;
```

6. **Создание страниц и компонентов**
   - **Home.js:**

```
import React from 'react';

function Home() {
    return (
        <div>
            <h1>Home Page</h1>
            <p>Welcome to the home page of our application.</p>
        </div>
    );
}

export default Home;
```

   - **About.js:**

```
import React from 'react';

function About() {
    return (
        <div>
            <h1>About Page</h1>
            <p>This is the about page of our application.</p>
        </div>
    );
}

export default About;
```

   - **Contact.js:**

```
import React from 'react';

function Contact() {
    return (
        <div>
```

```
            <h1>Contact Page</h1>
            <p>This is the contact page of our application.</p>
        </div>
    );
}

export default Contact;
```

- **Navbar.js:**

```
import React from 'react';
import { Link } from 'react-router-dom';

function Navbar() {
    return (
        <nav>
            <ul>
                <li><Link to="/">Home</Link></li>
                <li><Link to="/about">About</Link></li>
                <li><Link to="/contact">Contact</Link></li>
            </ul>
        </nav>
    );
}

export default Navbar;
```

7. **Заключение**
   - **Рефакторинг кода:**
     - Проверьте, что все компоненты организованы и структурированы.
     - Убедитесь, что нет дублирующегося кода и все компоненты переиспользуются.
   - **Тестирование и отладка:**
     - Проверьте работу всех компонентов и их взаимодействие.
     - Убедитесь, что приложение работает без ошибок и предупреждений в консоли.
   - **Завершение работы:**
     - Сохраните все изменения и закройте проект.

Лабораторная работа 8: Тестирование компонентов и приложений на React

Цель лабораторной работы:

Научиться писать модульные и интеграционные тесты для React-компонентов и приложений с использованием библиотеки Jest.

Описание работы:

1. **Установка и настройка Jest**
    - **Установка Jest и React Testing Library:**
        - В командной строке выполните команду для установки необходимых библиотек:

```
npm install --save-dev jest @testing-library/react @testing-library/jest-dom
```

    - **Настройка тестовой среды:**
        - В файле `package.json` добавьте следующую настройку для Jest:

```
json

"scripts": {
    "test": "jest"
}
```

2. **Написание модульных тестов**
    - **Создание тестов для простого компонента:**
        - В директории `src` создайте файл `Button.js` (если его еще нет):

```
import React from 'react';

function Button({ label, onClick }) {
    return <button onClick={onClick}>{label}</button>;
}

export default Button;
```

        - В директории `src` создайте папку `__tests__` и файл `Button.test.js`.
        - Добавьте в `Button.test.js` следующий код для тестирования компонента `Button`:

```
import React from 'react';
import { render, fireEvent } from '@testing-library/react';
import '@testing-library/jest-dom/extend-expect';
import Button from '../Button';

test('renders Button component with label', () => {
    const { getByText } = render(<Button label="Click Me" />);
    expect(getByText('Click Me')).toBeInTheDocument();
});

test('calls onClick handler when clicked', () => {
    const handleClick = jest.fn();
    const { getByText } = render(<Button label="Click Me"
onClick={handleClick} />);
    fireEvent.click(getByText('Click Me'));
    expect(handleClick).toHaveBeenCalledTimes(1);
});
```

- **Запуск тестов:**
  - В командной строке выполните команду для запуска тестов:

```
npm test
```

3. **Написание интеграционных тестов**
   - **Создание тестов для компонента с состоянием:**
     - В директории `src` создайте файл `Counter.js` (если его еще нет):

```
import React, { useState } from 'react';

function Counter() {
    const [count, setCount] = useState(0);

    return (
        <div>
            <p>Current count: {count}</p>
            <button onClick={() => setCount(count +
1)}>Increment</button>
            <button onClick={() => setCount(count -
1)}>Decrement</button>
        </div>
    );
}

export default Counter;
```

     - В директории `src/__tests__` создайте файл `Counter.test.js`.
     - Добавьте в `Counter.test.js` следующий код для тестирования компонента `Counter`:

```
import React from 'react';
import { render, fireEvent } from '@testing-library/react';
import '@testing-library/jest-dom/extend-expect';
import Counter from '../Counter';
test('increments and decrements the count', () => {
    const { getByText } = render(<Counter />);
    const incrementButton = getByText('Increment');
    const decrementButton = getByText('Decrement');
    const count = getByText('Current count: 0');

    fireEvent.click(incrementButton);
    expect(count).toHaveTextContent('Current count: 1');

    fireEvent.click(decrementButton);
    expect(count).toHaveTextContent('Current count: 0');
});
```

4. **Тестирование асинхронного кода**
   - **Создание тестов для компонента с запросом данных:**
     - В директории `src` создайте файл `DataFetcher.js` (если его еще нет):

```
import React, { useState, useEffect } from 'react';
```

```
function DataFetcher() {
    const [data, setData] = useState(null);
    const [loading, setLoading] = useState(true);

    useEffect(() => {
        fetch('https://jsonplaceholder.typicode.com/posts/1')
            .then(response => response.json())
            .then(data => {
                setData(data);
                setLoading(false);
            });
    }, []);

    if (loading) {
        return <p>Loading...</p>;
    }

    return (
        <div>
            <h3>{data.title}</h3>
            <p>{data.body}</p>
        </div>
    );
}

export default DataFetcher;
```

- В директории `src/__tests__` создайте файл `DataFetcher.test.js`.
- Добавьте в `DataFetcher.test.js` следующий код для тестирования компонента `DataFetcher`:

```
import React from 'react';
import { render, waitFor } from '@testing-library/react';
import '@testing-library/jest-dom/extend-expect';
import DataFetcher from '../DataFetcher';

global.fetch = jest.fn(() =>
    Promise.resolve({
        json: () => Promise.resolve({ title: 'Test Post', body: 'This
is a test post.' }),
    })
);

test('fetches and displays data', async () => {
    const { getByText } = render(<DataFetcher />);
    expect(getByText('Loading...')).toBeInTheDocument();

    await waitFor(() => expect(getByText('Test
Post')).toBeInTheDocument());
    expect(getByText('This is a test post.')).toBeInTheDocument();
});
```

5. **Заключение**
    - **Рефакторинг кода:**
        - Проверьте, что все тесты организованы и структурированы.

        - Убедитесь, что тесты покрывают все основные функциональности компонентов.
    - **Тестирование и отладка:**
        - Запустите все тесты и убедитесь, что они проходят без ошибок.
        - Убедитесь, что приложение работает без ошибок и предупреждений в консоли.
    - **Завершение работы:**
        - Сохраните все изменения и закройте проект.

Лабораторная работа 9: Разработка SPA на React

Цель лабораторной работы:

Практически освоить разработку Single Page Applications (SPA) на React, организовать структуру приложения и управлять его состоянием.

Описание работы:

1. **Создание структуры проекта**
    - **Инициализация проекта:**
        - Создайте новый проект с помощью Create React App, если он еще не создан:

```
npx create-react-app spa-project
cd spa-project
```

        - Установите необходимые зависимости, если их нет:

```
npm install react-router-dom
```

2. **Организация структуры приложения**
    - **Создание директорий и файлов:**
        - Создайте следующие директории и файлы для организации вашего проекта:

```
src/
├── components/
│   ├── Header.js
│   ├── Footer.js
```

```
├── pages/
│   ├── Home.js
│   ├── About.js
│   ├── Contact.js
│   ├── Posts.js
│   └── Post.js
├── services/
│   └── api.js
└── App.js
```

3. **Создание компонентов и страниц**
   - **Header.js:**

```
import React from 'react';
import { Link } from 'react-router-dom';

function Header() {
    return (
        <header>
            <nav>
                <ul>
                    <li><Link to="/">Home</Link></li>
                    <li><Link to="/about">About</Link></li>
                    <li><Link to="/contact">Contact</Link></li>
                    <li><Link to="/posts">Posts</Link></li>
                </ul>
            </nav>
        </header>
    );
}

export default Header;
```

   - **Footer.js:**

```
import React from 'react';

function Footer() {
    return (
        <footer>
            <p>&copy; 2024 SPA Project. All rights reserved.</p>
        </footer>
    );
}

export default Footer;
```

   - **Home.js:**

```
import React from 'react';

function Home() {
    return (
```

```
        <div>
            <h1>Home Page</h1>
            <p>Welcome to the home page of our SPA project.</p>
        </div>
    );
}

export default Home;
```

- **About.js:**

```
import React from 'react';

function About() {
    return (
        <div>
            <h1>About Page</h1>
            <p>This is the about page of our SPA project.</p>
        </div>
    );
}

export default About;
```

- **Contact.js:**

```
import React from 'react';

function Contact() {
    return (
        <div>
            <h1>Contact Page</h1>
            <p>This is the contact page of our SPA project.</p>
        </div>
    );
}

export default Contact;
```

- **Posts.js:**

```
import React, { useEffect, useState } from 'react';
import { Link } from 'react-router-dom';
import { fetchPosts } from '../services/api';

function Posts() {
    const [posts, setPosts] = useState([]);

    useEffect(() => {
        fetchPosts().then(data => setPosts(data));
    }, []);

    return (
        <div>
            <h1>Posts Page</h1>
            <ul>
```

```
                {posts.map(post => (
                    <li key={post.id}>
                        <Link to={`/post/${post.id}`}>{post.title}</Link>
                    </li>
                ))}
            </ul>
        </div>
    );
}

export default Posts;
```

- **Post.js:**

```
import React, { useEffect, useState } from 'react';
import { useParams } from 'react-router-dom';
import { fetchPost } from '../services/api';

function Post() {
    const { id } = useParams();
    const [post, setPost] = useState(null);

    useEffect(() => {
        fetchPost(id).then(data => setPost(data));
    }, [id]);

    if (!post) {
        return <p>Loading...</p>;
    }

    return (
        <div>
            <h1>{post.title}</h1>
            <p>{post.body}</p>
        </div>
    );
}

export default Post;
```

4. **Создание сервиса для API запросов**
   - **api.js:**

```
export const fetchPosts = async () => {
    const response = await
fetch('https://jsonplaceholder.typicode.com/posts');
    const data = await response.json();
    return data;
};

export const fetchPost = async (id) => {
    const response = await
fetch(`https://jsonplaceholder.typicode.com/posts/${id}`);
    const data = await response.json();
    return data;
};
```

5. **Настройка маршрутизации в приложении**

- **App.js:**

```
import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';
import Header from './components/Header';
import Footer from './components/Footer';
import Home from './pages/Home';
import About from './pages/About';
import Contact from './pages/Contact';
import Posts from './pages/Posts';
import Post from './pages/Post';

function App() {
    return (
        <Router>
            <div className="App">
                <Header />
                <Switch>
                    <Route path="/" exact component={Home} />
                    <Route path="/about" component={About} />
                    <Route path="/contact" component={Contact} />
                    <Route path="/posts" component={Posts} />
                    <Route path="/post/:id" component={Post} />
                </Switch>
                <Footer />
            </div>
        </Router>
    );
}

export default App;
```

6. **Управление состоянием приложения**
   - **Использование React Context для управления состоянием:**
     - В директории `src` создайте файл `AppContext.js`.
     - Добавьте в `AppContext.js` следующий код:

```
import React, { createContext, useState } from 'react';

export const AppContext = createContext();

export const AppProvider = ({ children }) => {
    const [state, setState] = useState({
        user: null,
        theme: 'light'
    });
    return (
        <AppContext.Provider value={{ state, setState }}>
            {children}
        </AppContext.Provider>
    );
};
```

   - **Оборачивание приложения в AppProvider:**
     - Обновите файл `App.js` для использования `AppProvider`:

```
import React from 'react';
```

```
import { BrowserRouter as Router, Route, Switch } from 'react-router-
dom';
import Header from './components/Header';
import Footer from './components/Footer';
import Home from './pages/Home';
import About from './pages/About';
import Contact from './pages/Contact';
import Posts from './pages/Posts';
import Post from './pages/Post';
import { AppProvider } from './AppContext';

function App() {
    return (
        <AppProvider>
            <Router>
                <div className="App">
                    <Header />
                    <Switch>
                        <Route path="/" exact component={Home} />
                        <Route path="/about" component={About} />
                        <Route path="/contact" component={Contact} />
                        <Route path="/posts" component={Posts} />
                        <Route path="/post/:id" component={Post} />
                    </Switch>
                    <Footer />
                </div>
            </Router>
        </AppProvider>
    );
}

export default App;
```

- **Использование контекста в компонентах:**
  - В любом компоненте вы можете получить доступ к состоянию и методам `AppContext` следующим образом:

```
import React, { useContext } from 'react';
import { AppContext } from '../AppContext';

function SomeComponent() {
    const { state, setState } = useContext(AppContext);

    const toggleTheme = () => {
        setState(prevState => ({
            ...prevState,
            theme: prevState.theme === 'light' ? 'dark' : 'light'
        }));
    };
    return (
        <div>
            <p>Current theme: {state.theme}</p>
            <button onClick={toggleTheme}>Toggle Theme</button>
        </div>
    );
}

export default SomeComponent;
```

7. **Заключение**
    - **Рефакторинг кода:**
        - Проверьте, что все компоненты организованы и структурированы.
        - Убедитесь, что нет дублирующегося кода и все компоненты переиспользуются.
    - **Тестирование и отладка:**
        - Проверьте работу всех компонентов и их взаимодействие.
        - Убедитесь, что приложение работает без ошибок и предупреждений в консоли.
    - **Завершение работы:**
        - Сохраните все изменения и закройте проект.

Лабораторная работа 10: Работа с контекстом и хуками в React

Цель лабораторной работы:

Изучить контекст и пользовательские хуки в React и применить их для передачи данных между компонентами и управления состоянием.

Описание работы:

1. **Создание и использование контекста в React**
    - **Создание контекста:**
        - В директории `src` создайте файл `AppContext.js`.
        - Добавьте в `AppContext.js` следующий код:

```
import React, { createContext, useState } from 'react';

export const AppContext = createContext();

export const AppProvider = ({ children }) => {
    const [state, setState] = useState({
        user: null,
        theme: 'light'
    });

    return (
        <AppContext.Provider value={{ state, setState }}>
            {children}
        </AppContext.Provider>
    );
};
```

    - **Оборачивание приложения в AppProvider:**
        - Обновите файл `App.js` для использования `AppProvider`:

```
import React from 'react';
```

```
import { BrowserRouter as Router, Route, Switch } from 'react-router-
dom';
import Header from './components/Header';
import Footer from './components/Footer';
import Home from './pages/Home';
import About from './pages/About';
import Contact from './pages/Contact';
import Posts from './pages/Posts';
import Post from './pages/Post';
import { AppProvider } from './AppContext';

function App() {
    return (
        <AppProvider>
            <Router>
                <div className="App">
                    <Header />
                    <Switch>
                        <Route path="/" exact component={Home} />
                        <Route path="/about" component={About} />
                        <Route path="/contact" component={Contact} />
                        <Route path="/posts" component={Posts} />
                        <Route path="/post/:id" component={Post} />
                    </Switch>
                    <Footer />
                </div>
            </Router>
        </AppProvider>
    );
}

export default App;
```

- **Использование контекста в компонентах:**
  - В любом компоненте вы можете получить доступ к состоянию и методам `AppContext` следующим образом:

```
import React, { useContext } from 'react';
import { AppContext } from '../AppContext';

function SomeComponent() {
    const { state, setState } = useContext(AppContext);

    const toggleTheme = () => {
        setState(prevState => ({
            ...prevState,
            theme: prevState.theme === 'light' ? 'dark' : 'light'
        }));
    };
    return (
        <div>
            <p>Current theme: {state.theme}</p>
            <button onClick={toggleTheme}>Toggle Theme</button>
        </div>
    );
}

export default SomeComponent;
```

2. **Создание пользовательских хуков**
    - **Создание пользовательского хука для управления темой:**
        - В директории `src` создайте файл `useTheme.js`.
        - Добавьте в `useTheme.js` следующий код:

```
import { useContext } from 'react';
import { AppContext } from './AppContext';

const useTheme = () => {
    const { state, setState } = useContext(AppContext);

    const toggleTheme = () => {
        setState(prevState => ({
            ...prevState,
            theme: prevState.theme === 'light' ? 'dark' : 'light'
        }));
    };

    return {
        theme: state.theme,
        toggleTheme
    };
};

export default useTheme;
```

    - **Использование пользовательского хука в компонентах:**
        - В любом компоненте вы можете использовать созданный хук следующим образом:

```
import React from 'react';
import useTheme from '../useTheme';

function ThemeToggle() {
    const { theme, toggleTheme } = useTheme();

    return (
        <div>
            <p>Current theme: {theme}</p>
            <button onClick={toggleTheme}>Toggle Theme</button>
        </div>
    );
}

export default ThemeToggle;
```

    - **Обновление App.js для использования компонента `ThemeToggle`:**
        - Откройте файл `App.js`.
        - Импортируйте компонент `ThemeToggle` и добавьте его в JSX:

```
import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-
dom';
import Header from './components/Header';
import Footer from './components/Footer';
import Home from './pages/Home';
```

```
import About from './pages/About';
import Contact from './pages/Contact';
import Posts from './pages/Posts';
import Post from './pages/Post';
import { AppProvider } from './AppContext';
import ThemeToggle from './components/ThemeToggle';

function App() {
    return (
        <AppProvider>
            <Router>
                <div className="App">
                    <Header />
                    <Switch>
                        <Route path="/" exact component={Home} />
                        <Route path="/about" component={About} />
                        <Route path="/contact" component={Contact} />
                        <Route path="/posts" component={Posts} />
                        <Route path="/post/:id" component={Post} />
                    </Switch>
                    <ThemeToggle />
                    <Footer />
                </div>
            </Router>
        </AppProvider>
    );
}

export default App;
```

3. **Передача данных между компонентами с использованием контекста**
    - **Создание компонента для управления пользователями:**
        - В директории `src` создайте файл `UserContext.js`.
        - Добавьте в `UserContext.js` следующий код:

```
import React, { createContext, useState } from 'react';

export const UserContext = createContext();

export const UserProvider = ({ children }) => {
    const [user, setUser] = useState(null);
    const login = (userData) => {
        setUser(userData);
    };
    const logout = () => {
        setUser(null);
    };
    return (
        <UserContext.Provider value={{ user, login, logout }}>
            {children}
        </UserContext.Provider>
    );
};
```

    - **Оборачивание приложения в UserProvider:**
        - Обновите файл `App.js` для использования `UserProvider`:

```
import React from 'react';
```

```
import { BrowserRouter as Router, Route, Switch } from 'react-router-
dom';
import Header from './components/Header';
import Footer from './components/Footer';
import Home from './pages/Home';
import About from './pages/About';
import Contact from './pages/Contact';
import Posts from './pages/Posts';
import Post from './pages/Post';
import { AppProvider } from './AppContext';
import { UserProvider } from './UserContext';
import ThemeToggle from './components/ThemeToggle';

function App() {
    return (
        <AppProvider>
            <UserProvider>
                <Router>
                    <div className="App">
                        <Header />
                        <Switch>
                            <Route path="/" exact component={Home} />
                            <Route path="/about" component={About} />
                            <Route path="/contact"
component={Contact} />
                            <Route path="/posts" component={Posts} />
                            <Route path="/post/:id" component={Post}
/>
                        </Switch>
                        <ThemeToggle />
                        <Footer />
                    </div>
                </Router>
            </UserProvider>
        </AppProvider>
    );
}

export default App;
```

- **Использование контекста пользователя в компонентах:**
  - В любом компоненте вы можете получить доступ к состоянию и методам `UserContext` следующим образом:

```
import React, { useContext } from 'react';
import { UserContext } from '../UserContext';

function UserProfile() {
    const { user, login, logout } = useContext(UserContext);

    return (
        <div>
            {user ? (
                <div>
                    <p>Welcome, {user.name}</p>
                    <button onClick={logout}>Logout</button>
                </div>
            ) : (
                <button onClick={() => login({ name: 'John Doe'
})}>Login</button>
```

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

export default UserProfile;
```

- **Обновление App.js для использования компонента `UserProfile`:**
  - Откройте файл `App.js`.
  - Импортируйте компонент `UserProfile` и добавьте его в JSX:

```
import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-
dom';
import Header from './components/Header';
import Footer from './components/Footer';
import Home from './pages/Home';
import About from './pages/About';
import Contact from './pages/Contact';
import Posts from './pages/Posts';
import Post from './pages/Post';
import { AppProvider } from './AppContext';
import { UserProvider } from './UserContext';
import ThemeToggle from './components/ThemeToggle';
import UserProfile from './components/UserProfile';

function App() {
    return (
        <AppProvider>
            <UserProvider>
                <Router>
                    <div className="App">
                        <Header />
                        <Switch>
                            <Route path="/" exact component={Home} />
                            <Route path="/about" component={About} />
                            <Route path="/contact"
component={Contact} />
                            <Route path="/posts" component={Posts} />
                            <Route path="/post/:id" component={Post}
/>
                        </Switch>
                        <ThemeToggle />
                        <UserProfile />
                        <Footer />
                    </div>
                </Router>
            </UserProvider>
        </AppProvider>
    );
}

export default App;
```

4. **Заключение**
   - **Рефакторинг кода:**
     - Проверьте, что все компоненты организованы и структурированы.

        - Убедитесь, что нет дублирующегося кода и все компоненты переиспользуются.
    - **Тестирование и отладка:**
        - Проверьте работу всех компонентов и их взаимодействие.
        - Убедитесь, что приложение работает без ошибок и предупреждений в консоли.
    - **Завершение работы:**
        - Сохраните все изменения и закройте проект.

Лабораторная работа 11: Архитектура приложений на React

Цель лабораторной работы:

Разработать приложение с использованием лучших практик и паттернов, организовать его архитектуру и структуру компонентов.

Описание работы:

1. **Создание структуры проекта**
    - **Инициализация проекта:**
        - Создайте новый проект с помощью Create React App, если он еще не создан:

```
npx create-react-app architecture-project
cd architecture-project
```

        - Установите необходимые зависимости:

```
npm install react-router-dom
```

    - **Организация структуры приложения:**
        - Создайте следующие директории и файлы для организации вашего проекта:

```
src/
├── components/
```

```
│   ├── common/
│   │   ├── Button.js
│   │   ├── Input.js
│   │   └── Navbar.js
│   ├── layout/
│   │   ├── Header.js
│   │   ├── Footer.js
│   │   └── Sidebar.js
├── pages/
│   ├── Home.js
│   ├── About.js
│   ├── Contact.js
│   ├── Posts.js
│   └── Post.js
├── services/
│   └── api.js
├── contexts/
│   └── AppContext.js
├── hooks/
│   └── useTheme.js
├── utils/
│   └── helpers.js
└── App.js
```

2. **Создание общих компонентов**
   - **Button.js:**

```
import React from 'react';
import PropTypes from 'prop-types';

function Button({ label, onClick }) {
    return <button onClick={onClick}>{label}</button>;
}

Button.propTypes = {
    label: PropTypes.string.isRequired,
    onClick: PropTypes.func.isRequired,
};

export default Button;
```

   - **Input.js:**

```
import React from 'react';
import PropTypes from 'prop-types';

function Input({ type, value, onChange }) {
    return <input type={type} value={value} onChange={onChange} />;
}

Input.propTypes = {
    type: PropTypes.string.isRequired,
    value: PropTypes.string.isRequired,
    onChange: PropTypes.func.isRequired,
};
```

```
export default Input;
```

- **Navbar.js:**

```
import React from 'react';
import { Link } from 'react-router-dom';

function Navbar() {
    return (
        <nav>
            <ul>
                <li><Link to="/">Home</Link></li>
                <li><Link to="/about">About</Link></li>
                <li><Link to="/contact">Contact</Link></li>
                <li><Link to="/posts">Posts</Link></li>
            </ul>
        </nav>
    );
}

export default Navbar;
```

3. **Создание компонентов макета**
   - **Header.js:**

```
import React from 'react';
import Navbar from '../common/Navbar';

function Header() {
    return (
        <header>
            <h1>My Application</h1>
            <Navbar />
        </header>
    );
}

export default Header;
```

   - **Footer.js:**

```
import React from 'react';
function Footer() {
    return (
        <footer>
            <p>&copy; 2024 My Application. All rights reserved.</p>
        </footer>
    );
}

export default Footer;
```

   - **Sidebar.js:**

```
import React from 'react';
```

```
function Sidebar() {
    return (
        <aside>
            <h2>Sidebar</h2>
            <p>This is the sidebar content.</p>
        </aside>
    );
}

export default Sidebar;
```

4. **Создание страниц приложения**
   - **Home.js:**

```
import React from 'react';

function Home() {
    return (
        <div>
            <h1>Home Page</h1>
            <p>Welcome to the home page of our application.</p>
        </div>
    );
}

export default Home;
```

   - **About.js:**

```
jsx

import React from 'react';

function About() {
    return (
        <div>
            <h1>About Page</h1>
            <p>This is the about page of our application.</p>
        </div>
    );
}

export default About;
```

   - **Contact.js:**

```
import React from 'react';

function Contact() {
    return (
        <div>
            <h1>Contact Page</h1>
            <p>This is the contact page of our application.</p>
        </div>
    );
}
export default Contact;
```

- **Posts.js:**

```
import React, { useEffect, useState } from 'react';
import { Link } from 'react-router-dom';
import { fetchPosts } from '../services/api';

function Posts() {
    const [posts, setPosts] = useState([]);

    useEffect(() => {
        fetchPosts().then(data => setPosts(data));
    }, []);

    return (
        <div>
            <h1>Posts Page</h1>
            <ul>
                {posts.map(post => (
                    <li key={post.id}>
                        <Link to={`/post/${post.id}`}>{post.title}</Link>
                    </li>
                ))}
            </ul>
        </div>
    );
}

export default Posts;
```

- **Post.js:**

```
jsx

import React, { useEffect, useState } from 'react';
import { useParams } from 'react-router-dom';
import { fetchPost } from '../services/api';

function Post() {
    const { id } = useParams();
    const [post, setPost] = useState(null);

    useEffect(() => {
        fetchPost(id).then(data => setPost(data));
    }, [id]);

    if (!post) {
        return <p>Loading...</p>;
    }
    return (
        <div>
            <h1>{post.title}</h1>
            <p>{post.body}</p>
        </div>
    );
}
export default Post;
```

5. **Создание сервиса для API запросов**

- **api.js:**

```
export const fetchPosts = async () => {
    const response = await
fetch('https://jsonplaceholder.typicode.com/posts');
    const data = await response.json();
    return data;
};

export const fetchPost = async (id) => {
    const response = await
fetch(`https://jsonplaceholder.typicode.com/posts/${id}`);
    const data = await response.json();
    return data;
};
```

6. **Создание контекста и хуков**
   - **AppContext.js:**

```
import React, { createContext, useState } from 'react';

export const AppContext = createContext();

export const AppProvider = ({ children }) => {
    const [state, setState] = useState({
        user: null,
        theme: 'light'
    });

    return (
        <AppContext.Provider value={{ state, setState }}>
            {children}
        </AppContext.Provider>
    );
};
```

   - **useTheme.js:**

```
import { useContext } from 'react';
import { AppContext } from '../contexts/AppContext';

const useTheme = () => {
    const { state, setState } = useContext(AppContext);
    const toggleTheme = () => {
        setState(prevState => ({
            ...prevState,
            theme: prevState.theme === 'light' ? 'dark' : 'light'
        }));
    };
    return {
        theme: state.theme,
        toggleTheme
    };
};

export default useTheme;
```

7. **Настройка маршрутизации в приложении**
   - **App.js:**

```
import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';
import Header from './components/layout/Header';
import Footer from './components/layout/Footer';
import Home from './pages/Home';
import About from './pages/About';
import Contact from './pages/Contact';
import Posts from './pages/Posts';
import Post from './pages/Post';
import { AppProvider } from './contexts/AppContext';
import Sidebar from './components/layout/Sidebar';
import ThemeToggle from './components/ThemeToggle';
import UserProfile from './components/UserProfile';

function App() {
    return (
        <AppProvider>
            <Router>
                <div className="App">
                    <Header />
                    <Sidebar />
                    <Switch>
                        <Route path="/" exact component={Home} />
                        <Route path="/about" component={About} />
                        <Route path="/contact" component={Contact} />
                        <Route path="/posts" component={Posts} />
                        <Route path="/post/:id" component={Post} />
                    </Switch>
                    <ThemeToggle />
                    <UserProfile />
                    <Footer />
                </div>
            </Router>
        </AppProvider>
    );
}

export default App;
```

8. **Использование контекста и хуков в компонентах**
   - **ThemeToggle.js:**

```
import React from 'react';
import useTheme from '../hooks/useTheme';

function ThemeToggle() {
    const { theme, toggleTheme } = useTheme();
    return (
        <div>
            <p>Current theme: {theme}</p>
            <button onClick={toggleTheme}>Toggle Theme</button>
        </div>
    );
}

export default ThemeToggle;
```

- **UserProfile.js:**

```
import React, { useContext } from 'react';
import { AppContext } from '../contexts/AppContext';

function UserProfile() {
    const { state, setState } = useContext(AppContext);

    const login = () => {
        setState(prevState => ({
            ...prevState,
            user: { name: 'John Doe' }
        }));
    };

    const logout = () => {
        setState(prevState => ({
            ...prevState,
            user: null
        }));
    };

    return (
        <div>
            {state.user ? (
                <div>
                    <p>Welcome, {state.user.name}</p>
                    <button onClick={logout}>Logout</button>
                </div>
            ) : (
                <button onClick={login}>Login</button>
            )}
        </div>
    );
}

export default UserProfile;
```

9. **Заключение**
   - **Рефакторинг кода:**
     - Проверьте, что все компоненты организованы и структурированы.
     - Убедитесь, что нет дублирующегося кода и все компоненты переиспользуются.
   - **Тестирование и отладка:**
     - Проверьте работу всех компонентов и их взаимодействие.
     - Убедитесь, что приложение работает без ошибок и предупреждений в консоли.
   - **Завершение работы:**
     - Сохраните все изменения и закройте проект.

Лабораторная работа 12: SSR и применение в React

Цель лабораторной работы:

Изучить Server-Side Rendering (SSR) и его применение в React-приложениях для улучшения производительности и SEO.

Описание работы:

1. **Создание проекта с Next.js**
   - **Инициализация проекта:**
     - Создайте новый проект с помощью Create Next App:

```
npx create-next-app ssr-project
cd ssr-project
```

   - **Запуск проекта:**
     - В командной строке выполните команду для запуска проекта:

```
npm run dev
```

2. **Создание структуры проекта**
   - **Организация структуры приложения:**
     - В директории `pages` создайте следующие файлы:

```
pages/
├── index.js
├── about.js
├── contact.js
├── posts/
│   ├── index.js
│   └── [id].js
```

3. **Создание страниц с SSR**
   - **index.js:**

```
import React from 'react';
import Link from 'next/link';

function Home({ posts }) {
    return (
        <div>
            <h1>Home Page</h1>
            <ul>
                {posts.map(post => (
                    <li key={post.id}>
                        <Link href={`/posts/${post.id}`}>
                            <a>{post.title}</a>
                        </Link>
```

```
                    </li>
                ))}
            </ul>
        </div>
    );
}

export async function getServerSideProps() {
    const res = await fetch('https://jsonplaceholder.typicode.com/posts');
    const posts = await res.json();
    return {
        props: {
            posts,
        },
    };
}

export default Home;
```

- **about.js:**

```
import React from 'react';

function About() {
    return (
        <div>
            <h1>About Page</h1>
            <p>This is the about page of our application.</p>
        </div>
    );
}

export default About;
```

- **contact.js:**

```
import React from 'react';

function Contact() {
    return (
        <div>
            <h1>Contact Page</h1>
            <p>This is the contact page of our application.</p>
        </div>
    );
}

export default Contact;
```

- **posts/index.js:**

```
import React from 'react';
import Link from 'next/link';

function Posts({ posts }) {
    return (
        <div>
```

```
            <h1>Posts Page</h1>
            <ul>
                {posts.map(post => (
                    <li key={post.id}>
                        <Link href={`/posts/${post.id}`}>
                            <a>{post.title}</a>
                        </Link>
                    </li>
                ))}
            </ul>
        </div>
    );
}

export async function getServerSideProps() {
    const res = await fetch('https://jsonplaceholder.typicode.com/posts');
    const posts = await res.json();
    return {
        props: {
            posts,
        },
    };
}

export default Posts;
```

- **posts/[id].js:**

```
import React from 'react';
import { useRouter } from 'next/router';

function Post({ post }) {
    const router = useRouter();
    const { id } = router.query;

    return (
        <div>
            <h1>{post.title}</h1>
            <p>{post.body}</p>
        </div>
    );
}

export async function getServerSideProps({ params }) {
    const res = await
fetch(`https://jsonplaceholder.typicode.com/posts/${params.id}`);
    const post = await res.json();
    return {
        props: {
            post,
        },
    };
}

export default Post;
```

4. **Использование компонентов для общих частей**
   - **Создание директории `components`:**
     - В корне проекта создайте директорию `components`.

- Внутри `components` создайте файл `Navbar.js`:

```
import React from 'react';
import Link from 'next/link';

function Navbar() {
    return (
        <nav>
            <ul>
                <li>
                    <Link href="/">
                        <a>Home</a>
                    </Link>
                </li>
                <li>
                    <Link href="/about">
                        <a>About</a>
                    </Link>
                </li>
                <li>
                    <Link href="/contact">
                        <a>Contact</a>
                    </Link>
                </li>
                <li>
                    <Link href="/posts">
                        <a>Posts</a>
                    </Link>
                </li>
            </ul>
        </nav>
    );
}

export default Navbar;
```

- **Использование `Navbar` в страницах:**
  - Обновите страницы, чтобы использовать компонент `Navbar`:

```
// pages/index.js (обновление)
import React from 'react';
import Link from 'next/link';
import Navbar from '../components/Navbar';

function Home({ posts }) {
    return (
        <div>
            <Navbar />
            <h1>Home Page</h1>
            <ul>
                {posts.map(post => (
                    <li key={post.id}>
                        <Link href={`/posts/${post.id}`}>
                            <a>{post.title}</a>
                        </Link>
                    </li>
                ))}
            </ul>
        </div>
```

```
    );
}

export async function getServerSideProps() {
    const res = await
fetch('https://jsonplaceholder.typicode.com/posts');
    const posts = await res.json();
    return {
        props: {
            posts,
        },
    };
}

export default Home;
```

```
// pages/about.js (обновление)
import React from 'react';
import Navbar from '../components/Navbar';

function About() {
    return (
        <div>
            <Navbar />
            <h1>About Page</h1>
            <p>This is the about page of our application.</p>
        </div>
    );
}

export default About;
```

```
// pages/contact.js (обновление)
import React from 'react';
import Navbar from '../components/Navbar';

function Contact() {
    return (
        <div>
            <Navbar />
            <h1>Contact Page</h1>
            <p>This is the contact page of our application.</p>
        </div>
    );
}

export default Contact;
```

```
// pages/posts/index.js (обновление)
import React from 'react';
import Link from 'next/link';
import Navbar from '../../components/Navbar';

function Posts({ posts }) {
    return (
        <div>
            <Navbar />
            <h1>Posts Page</h1>
            <ul>
                {posts.map(post => (
                    <li key={post.id}>
```

```
                        <Link href={`/posts/${post.id}`}>
                            <a>{post.title}</a>
                        </Link>
                    </li>
                ))}
            </ul>
        </div>
    );
}

export async function getServerSideProps() {
    const res = await
fetch('https://jsonplaceholder.typicode.com/posts');
    const posts = await res.json();
    return {
        props: {
            posts,
        },
    };
}

export default Posts;
```

```
// pages/posts/[id].js (обновление)
import React from 'react';
import { useRouter } from 'next/router';
import Navbar from '../../components/Navbar';

function Post({ post }) {
    const router = useRouter();
    const { id } = router.query;

    return (
        <div>
            <Navbar />
            <h1>{post.title}</h1>
            <p>{post.body}</p>
        </div>
    );
}
```

```
export async function getServerSideProps({ params }) {
    const res = await
fetch(`https://jsonplaceholder.typicode.com/posts/${params.id}`);
    const post = await res.json();
    return {
        props: {
            post,
        },
    };
}

export default Post;
```

5. **Заключение**
    - **Рефакторинг кода:**

        - Проверьте, что все компоненты организованы и структурированы.
        - Убедитесь, что нет дублирующегося кода и все компоненты переиспользуются.
    - **Тестирование и отладка:**
        - Проверьте работу всех компонентов и их взаимодействие.
        - Убедитесь, что приложение работает без ошибок и предупреждений в консоли.
    - **Завершение работы:**
        - Сохраните все изменения и закройте проект.

Приложение 2

КОМПЛЕКТ ОЦЕНОЧНЫХ СРЕДСТВ К РУБЕЖНОМУ КОНТРОЛЮ

Первая рубежная аттестация (6 –й семестр)

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

Вариант № 1

Дисциплина «Разработка веб-приложений с применением фреймворков»

1. Какова основная цель использования библиотеки React в разработке веб-приложений?
2. Какие основные преимущества предоставляет JSX в сравнении с обычным JavaScript?

Преподаватель /М.К. Абдулаев/

Зав. кафедрой /Л.Р. Магомаева

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

Вариант № 2

Дисциплина «Разработка веб-приложений с применением фреймворков»

1. Какие методы React используются для управления состоянием компонентов? Приведите примеры их применения.
2. Как осуществляется роутинг и навигация в веб-приложениях на React? Какие инструменты для этого используются?

Преподаватель /М.К. Абдулаев/

Зав. кафедрой /Л.Р. Магомаева

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

Вариант № 3

Дисциплина «Разработка веб-приложений с применением фреймворков»

1. Как происходит стилизация компонентов в React-приложениях? Расскажите о преимуществах подходов CSS Modules и Styled Components.
2. Как обрабатываются формы и события в React-приложениях? Как организовать валидацию форм?

Преподаватель /М.К. Абдулаев/

Зав. кафедрой /Л.Р. Магомаева

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

Вариант № 4
Дисциплина «Разработка веб-приложений с применением фреймворков»

1. Как осуществляется взаимодействие с сервером в React-приложениях? Укажите на основные методы и инструменты.
2. Какие механизмы используются для реализации аутентификации и авторизации в приложениях на React?

Преподаватель /М.К. Абдулаев/

Зав. кафедрой /Л.Р. Магомаева

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

Вариант № 5
Дисциплина «Разработка веб-приложений с применением фреймворков»

1. Какие средства и методы оптимизации производительности могут быть применены в React-приложениях?
2. Какие инструменты и подходы используются для тестирования компонентов и приложений на React?

Преподаватель /М.К. Абдулаев/

Зав. кафедрой /Л.Р. Магомаева

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

Вариант № 1
Дисциплина «Разработка веб-приложений с применением фреймворков»

1. Каковы особенности разработки Single Page Applications (SPA) на React? В чем их преимущества и недостатки?
2. Какие методы и инструменты используются для работы с контекстом и хуками в React-приложениях?

Преподаватель /М.К. Абдулаев/

Зав. кафедрой /Л.Р. Магомаева/

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

Вариант № 2
Дисциплина «Разработка веб-приложений с применением фреймворков»

1. Какие паттерны и лучшие практики применяются при проектировании архитектуры приложений на React?
2. Как SSR (Server-Side Rendering) влияет на производительность веб-приложений на React? В каких случаях его стоит использовать?

Преподаватель /М.К. Абдулаев/

Зав. кафедрой /Л.Р. Магомаева/

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

Вариант № 3
Дисциплина «Разработка веб-приложений с применением фреймворков»

1. Что такое Next.js и какие возможности он предоставляет для разработки приложений на React?
2. Какие средства и методы используются для тестирования компонентов и приложений на React?

Преподаватель /М.К. Абдулаев/
Зав. кафедрой /Л.Р. Магомаева/

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

Вариант № 4
Дисциплина «Разработка веб-приложений с применением фреймворков»

1. Какие аспекты безопасности следует учитывать при разработке и аутентификации в приложениях на React?
2. Какие сценарии оптимизации производительности чаще всего применяются в веб-приложениях на React?

Преподаватель /М.К. Абдулаев/

Зав. кафедрой /Л.Р. Магомаева/

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

Вариант № 5
Дисциплина «Разработка веб-приложений с применением фреймворков»

1. Какие особенности работы с API следует учитывать при взаимодействии с сервером в React-приложениях?
2. Какие инструменты и методы используются для управления состоянием приложения в React и обеспечения его целостности?

Преподаватель /М.К. Абдулаев/

Зав. кафедрой /Л.Р. Магомаева/

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

Вариант № 1
Дисциплина «Разработка веб-приложений с применением фреймворков»

1. Какие основные принципы оптимизации производительности следует учитывать при разработке веб-приложений на React?
2. Какие инструменты и методы можно использовать для оценки производительности веб-приложений на React?

Преподаватель /М.К. Абдулаев/

Зав. кафедрой /Л.Р. Магомаева/

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

Вариант № 2
Дисциплина «Разработка веб-приложений с применением фреймворков»

1. Какие типичные проблемы производительности могут возникать в веб-приложениях на React и как их можно решить?
2. Какие сценарии тестирования компонентов и приложений на React вы считаете наиболее важными для обеспечения качества кода?

Преподаватель /М.К. Абдулаев/

Зав. кафедрой /Л.Р. Магомаева/

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

Вариант № 3
Дисциплина «Разработка веб-приложений с применением фреймворков»

1. Какие основные методы и инструменты используются для тестирования компонентов и приложений на React?
2. Какие преимущества и недостатки связаны с разработкой Single Page Applications (SPA) на React?

Преподаватель /М.К. Абдулаев/

Зав. кафедрой /Л.Р. Магомаева/

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

Вариант № 4
Дисциплина «Разработка веб-приложений с применением фреймворков»

1. Какие особенности разработки SPA на React следует учитывать для обеспечения быстрой загрузки и отзывчивости приложения?
2. Как работает контекст в React и какие задачи можно решать с его помощью? Приведите примеры использования контекста.

Преподаватель /М.К. Абдулаев/

Зав. кафедрой /Л.Р. Магомаева/

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

Вариант № 5
Дисциплина «Разработка веб-приложений с применением фреймворков»

1. Какие хуки доступны в React и для каких задач они предназначены? Приведите примеры использования хуков.
2. Какие паттерны и лучшие практики применяются при проектировании архитектуры приложений на React?

Преподаватель /М.К. Абдулаев/

Зав. кафедрой /Л.Р. Магомаева/

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

Вариант № 1
Дисциплина «Разработка веб-приложений с применением фреймворков»

1. Как SSR (Server-Side Rendering) влияет на производительность и SEO веб-приложений на React? В каких случаях его стоит применять?
2. Какие основные преимущества предоставляет SSR в сравнении с клиентским рендерингом?

Преподаватель /М.К. Абдулаев/

Зав. кафедрой /Л.Р. Магомаева/

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

Вариант № 2
Дисциплина «Разработка веб-приложений с применением фреймворков»

1. Какие основные принципы и подходы следует учитывать при разработке архитектуры приложений на React с использованием SSR?
2. Какие ключевые особенности и возможности предоставляет фреймворк Next.js для разработки веб-приложений на React?

Преподаватель /М.К. Абдулаев/

Зав. кафедрой /Л.Р. Магомаева/

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

Вариант № 3
Дисциплина «Разработка веб-приложений с применением фреймворков»

1. Какие основные задачи можно решать с помощью Next.js и какие преимущества он предоставляет разработчикам?
2. Какие сценарии тестирования компонентов и приложений на React вы считаете наиболее важными для обеспечения качества кода?

Преподаватель /М.К. Абдулаев/

Зав. кафедрой /Л.Р. Магомаева/

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

Вариант № 4
Дисциплина «Разработка веб-приложений с применением фреймворков»

1. Какие средства и методы используются для тестирования компонентов и приложений на React?
2. Какие архитектурные паттерны и лучшие практики помогают обеспечить масштабируемость и поддерживаемость приложений на React?

Преподаватель /М.К. Абдулаев/

Зав. кафедрой /Л.Р. Магомаева/

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

Вариант № 5
Дисциплина «Разработка веб-приложений с применением фреймворков»

1. Какие сценарии оптимизации производительности чаще всего применяются в веб-приложениях на React?
2. Какие аспекты безопасности следует учитывать при разработке приложений на React с использованием SSR и Next.js?

Преподаватель /М.К. Абдулаев/

Зав. кафедрой /Л.Р. Магомаева/

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

БИЛЕТ № 1

Дисциплина «Разработка веб-приложений с применением фреймворков»

Институт _ИЦЭиТП_ специальность _ПИ__6__семестр

1. Какова основная цель использования библиотеки React в разработке веб-приложений?
2. Какие основные преимущества предоставляет JSX в сравнении с обычным JavaScript?
3. Какие методы React используются для управления состоянием компонентов? Приведите примеры их применения.
4. Как осуществляется роутинг и навигация в веб-приложениях на React? Какие инструменты для этого используются?

УТВЕРЖДЕНО зав. кафедрой
на заседании кафедры
протокол № ___ от _____ _____ Л.Р. Магомаева

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

БИЛЕТ № 2

Дисциплина «Разработка веб-приложений с применением фреймворков»

Институт _ИЦЭиТП_ специальность _ПИ__6__семестр

1. Как происходит стилизация компонентов в React-приложениях? Расскажите о преимуществах подходов CSS Modules и Styled Components.
2. Как обрабатываются формы и события в React-приложениях? Как организовать валидацию форм?
3. Как осуществляется взаимодействие с сервером в React-приложениях? Укажите на основные методы и инструменты.
4. Какие механизмы используются для реализации аутентификации и авторизации в приложениях на React?

УТВЕРЖДЕНО зав. кафедрой
на заседании кафедры
протокол № ___ от _____ _____ Л.Р. Магомаева

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

БИЛЕТ № 3

Дисциплина «Разработка веб-приложений с применением фреймворков»

Институт _ИЦЭиТП_ специальность _ПИ__6__семестр

1. Какие средства и методы оптимизации производительности могут быть применены в React-приложениях?
2. Какие инструменты и подходы используются для тестирования компонентов и приложений на React?
3. Каковы особенности разработки Single Page Applications (SPA) на React? В чем их преимущества и недостатки?
4. Какие методы и инструменты используются для работы с контекстом и хуками в React-приложениях?

УТВЕРЖДЕНО зав. кафедрой
на заседании кафедры
протокол № ___ от _____ _____ Л.Р. Магомаева

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

БИЛЕТ № 4

Дисциплина «Разработка веб-приложений с применением фреймворков»

Институт _ИЦЭиТП_ специальность _ПИ__6__семестр

1. Какие паттерны и лучшие практики применяются при проектировании архитектуры приложений на React?
2. Как SSR (Server-Side Rendering) влияет на производительность веб-приложений на React? В каких случаях его стоит использовать?
3. Что такое Next.js и какие возможности он предоставляет для разработки приложений на React?
4. Какие средства и методы используются для тестирования компонентов и приложений на React?

УТВЕРЖДЕНО зав. кафедрой
на заседании кафедры
протокол № ___ от _____ _____ Л.Р. Магомаева

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

БИЛЕТ № 5

Дисциплина «Разработка веб-приложений с применением фреймворков»

Институт _ИЦЭиТП_ специальность _ПИ__6__семестр

1. Какие аспекты безопасности следует учитывать при разработке и аутентификации в приложениях на React?
2. Какие сценарии оптимизации производительности чаще всего применяются в веб-приложениях на React?
3. Какие особенности работы с API следует учитывать при взаимодействии с сервером в React-приложениях?
4. Какие инструменты и методы используются для управления состоянием приложения в React и обеспечения его целостности?

УТВЕРЖДЕНО зав. кафедрой
на заседании кафедры
протокол № ___ от _____ _____ Л.Р. Магомаева

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

БИЛЕТ № 1

Дисциплина «Разработка веб-приложений с применением фреймворков»

Институт _ИЦЭиТП_ специальность _ПИ__7__семестр

1. Какие основные принципы оптимизации производительности следует учитывать при разработке веб-приложений на React?
2. Какие инструменты и методы можно использовать для оценки производительности веб-приложений на React?

УТВЕРЖДЕНО зав. кафедрой
на заседании кафедры
протокол № ___ от _____ _____ Л.Р. Магомаева

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

БИЛЕТ № 2

Дисциплина «Разработка веб-приложений с применением фреймворков»

Институт _ИЦЭиТП_ специальность _ПИ__7__семестр

1. Какие типичные проблемы производительности могут возникать в веб-приложениях на React и как их можно решить?
2. Какие сценарии тестирования компонентов и приложений на React вы считаете наиболее важными для обеспечения качества кода?

УТВЕРЖДЕНО зав. кафедрой
на заседании кафедры
протокол № ___ от _____ _____ Л.Р. Магомаева

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

БИЛЕТ № 3

Дисциплина «Разработка веб-приложений с применением фреймворков»
Институт _ИЦЭиТП_ специальность _ПИ__7__семестр

1. Какие основные методы и инструменты используются для тестирования компонентов и приложений на React?
2. Какие преимущества и недостатки связаны с разработкой Single Page Applications (SPA) на React?

УТВЕРЖДЕНО зав. кафедрой
на заседании кафедры
протокол № ___ от _____ _____ Л.Р. Магомаева

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

БИЛЕТ № 4

Дисциплина «Разработка веб-приложений с применением фреймворков»
Институт _ИЦЭиТП_ специальность _ПИ__7__семестр

1. Какие особенности разработки SPA на React следует учитывать для обеспечения быстрой загрузки и отзывчивости приложения?
2. Как работает контекст в React и какие задачи можно решать с его помощью? Приведите примеры использования контекста.

УТВЕРЖДЕНО зав. кафедрой
на заседании кафедры
протокол № ___ от _____ _____ Л.Р. Магомаева

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. акад. М.Д. Миллионщикова

БИЛЕТ № 5

Дисциплина «Разработка веб-приложений с применением фреймворков»
Институт _ИЦЭиТП_ специальность _ПИ__7__семестр

1. Какие хуки доступны в React и для каких задач они предназначены? Приведите примеры использования хуков.
2. Какие паттерны и лучшие практики применяются при проектировании архитектуры приложений на React?

УТВЕРЖДЕНО зав. кафедрой
на заседании кафедры
протокол № ___ от _____ _____ Л.Р. Магомаева