Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Ордена Трудового Красного Знамени

федеральное государственное бюджетное образовательное учреждение высшего образования

МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И ИНФОРМАТИКИ

Кафедра «Математической кибернетики и информационных технологий»

Отчет по проектному практикуму

Выполнил:

студент группы БВТ2201

Соколов.И.С

# 

# Выполнение

1. 1) Вот что просиходит в файле app: Импорты:

• В начале файла импортируются необходимые зависимости, такие как React компоненты и функции, иконки, пользовательский-хук useForm, а также функции getRandomChar и getSpecialChar из других файлов.

2. Компонент App:

• Компонент App создает и отображает интерфейс приложения.

• Используется хук useForm для управления состоянием формы. Этот хук инициализируется объектом values, который содержит параметры генерации пароля (длина, использование заглавных букв, строчных букв, цифр и специальных символов).

• Состояние result используется для хранения сгенерированного пароля.

3. Генерация пароля:

• В компоненте App определен массив fieldsArray, который содержит объекты, представляющие разные параметры генерации пароля (заглавные буквы, строчные буквы, цифры, специальные символы). Каждый объект имеет поле field (булевое значение, указывающее, включен ли данный параметр) и функцию getChar, которая возвращает случайный символ в соответствии с заданными диапазонами символов.

• Когда пользователь отправляет форму (нажимает "Generate Password"), вызывается функция handleOnSubmit. Она выполняет следующие шаги:

• Проверяет, есть ли хотя бы один выбранный параметр для генерации пароля.

• Если есть, то генерирует пароль, выбирая случайные символы в соответствии с выбранными параметрами и длиной пароля.

• Результат генерации сохраняется в состоянии result.

4. Копирование в буфер обмена:

• Функция handleClipboard вызывается, когда пользователь щелкает по значку буфера обмена (иконка "FaClipboard").

• Если у пользователя есть сгенерированный пароль, он копируется в буфер обмена с использованием navigator.clipboard.writeText. После успешного копирования выводится уведомление об успехе. В противном случае выводится уведомление об ошибке.

5. Отображение интерфейса:

• Возвращаемый JSX-код в компоненте App отображает форму с настройками параметров пароля, поле для отображения сгенерированного пароля и иконку для копирования пароля в буфер об-мена.

Компонент App обеспечивает весь функционал для генерации и отображения паролей с различными параметрами и удобного копирования их в буфер обмена.

2) Вот что происходит в файле main:

1. Импорты:

• В начале файла main.js импортируются необходимые зависимости и компоненты, которые будут использоваться в приложении.

2. createRoot и StrictMode:

• В файле используется функция createRoot из библиотеки react-dom/client. Эта функция предназначена для асинхронной инициализации приложения и улучшения производительности.

• Обертка <StrictMode> вокруг компонента App позволяет активировать строгий режим (strict mode), который помогает выявить потенциальные проблемы в приложении на ранних этапах разработки.

3. Инициализация корневого элемента:

• Сначала определяется корневой элемент с идентификатором 'root', на который будет внедрено приложение.

• Затем используется createRoot для создания корневого элемента приложения.

4. Рендеринг приложения:

• С помощью root.render(), компонент <App /> отрисовывается в корневом элементе.

• Компонент <Toaster /> также рендерится внутри строгого режима <StrictMode>. Этот компонент предоставляет возможность выводить уведомления в приложении, например, при успешном копировании пароля.

В целом, файл main.js отвечает за инициализацию React-приложения, создание корневого элемента и отображение компонента <App />, который пред-ставляет собой основную часть вашего парольного генератора.

3) Вот что просиходит в файле, который является кастомным хуком useForm:

1. Импорты:

• В файле useForm.js импортируется только функция useState из библиотеки React.

2. Определение пользовательского хука useForm:

• Функция useForm принимает один аргумент initialValues, который представляет собой объект с начальными значениями полей формы. Обычно это объект, где ключи соответствуют именам полей формы, а значения - начальным значениям этих полей.

3. Создание состояния и функции для управления формой:

• Внутри пользовательского хука useForm используется хук useState для создания состояния values, которое будет содержать значения полей формы. Начальное значение values устанавливается равным initialValues.

• Затем определяется функция setValues, которая принимает событие (e) в качестве аргумента и обновляет состояние values. Обновление зависит от типа элемента формы. Если элемент формы имеет тип 'checkbox', то будет обновляться его свойство checked, иначе - свойство value. Таким образом, функция setValues позволяет обновлять значения полей формы в зависимости от их типов.

4. Возврат пользовательского хука:

• После создания состояния и функции setValues, пользовательский хук useForm возвращает массив с двумя элементами:

• values: объект, содержащий текущие значения полей формы.

• setValues: функция для обновления значений полей формы.

Этот пользовательский хук useForm облегчает управление значениями полей формы в компоненте <App />. Он позволяет легко связывать значения полей с состоянием приложения и автоматически обновлять их при взаимодействии с пользователем.

4) 3) Вот что происходит в утилите utils:

1. getRandomChar(min, max):

• Эта функция принимает два аргумента, min и max, которые представляют собой диапазон символов в таблице Unicode. Например, min и max могут быть числами, представляющими диапазон от кода символа, соответствующего 'A' до кода символа, соответствующего 'Z', если вы хотите получить случайную заглавную букву.

• Функция генерирует случайное число в заданном диапазоне и использует его для создания символа, который соответствует этому числу в таблице Unicode. Этот символ представляет собой случайный символ в указанном диапазоне.

2. getSpecialChar():

• Эта функция возвращает случайный специальный символ из заданной строки specialChar. Внутри этой строки находятся раз-личные специальные символы, такие как !, ", #, $, %, и так далее.

• Функция генерирует случайный индекс в строке specialChar и возвращает символ, соответствующий этому индексу. Таким образом, она предоставляет случайный специальный символ из -за данного набора.

Эти две функции используются в компоненте <App /> для генерации случайных символов и специальных символов, которые составляют сгенерированный пароль с учетом выбранных параметров (заглавные буквы, строчные бук-вы, цифры и специальные символы). Функции getRandomChar и getSpecialChar способствуют разнообразию и безопасности паролей, созданных нашим приложением.

**Вывод:** был создан генератор паролей на основе React js + vite.js, который имеет несколько чекбоксов, копирование пароля в буфер обмена и уведомлениями об ошибке.