تکلیف شماره یک :

سوال یک : فایل package.json چیه و چه اطلاعاتی داخلش قرار میگیره، کلید های  
name, version, entry, private, dependencies, devDependencies, peerDependencies, scripts چی هستن و چه مقادیری دارن؟

جواب سوال 1-4 :

یک فایل با فرمت json هستش که یک سری اطلاعات را درمورد پروژه شامل میشه و مثلا باعث میشه npm بتونه پروژه رو مدیریت کنه و ... که تنها فایدش نیست

package.json یک فایل با فرمت جیسون هست که پس از یک سری key و value ها ساخته شده

name :

این key همون نام پروژمونه و اگه میخایم یک پکیج npm بسازیم و اون رو publish کنیم حتما قبل انتخاب اسم داخل سایت npm سرچ میکنیم که اسم مدنظرمون قبلا توسط کسی دیگه ای گرفته نشده باشه.

اگر هم پروژمون قرار نیست داخل npm منتشر بشه در این مورد اهمیتی نداره .

Version :

ورژن کنونی پروژه رو مشخص می کنه که درصورتی که میخایم پروژمون رو در npm منتشر کنیم این فیلد مهمه و هر بار که میخایم پکیجمونو در npm آپدیت کنیم باید ورژن بالاتری نسبت به ورژن قبلی داخل فایل package.json مون باشه.

Description :

این فیلد توضیحات مربوط به پروژمونه که بهتره داخلش توضیح بدیم چه کاری پروژمون انجام میده دقیقا چه چیزی هستش و درصورت انتشارآن به مردم کمک میکنه بهتر اون رو پیدا کنن.

Scripts :

این فیلد شامل یک سری اسکریپت هایی میشه که در زمان های مختلف میتونند اجرا بشوند

Entry : فایل اصلی پروژس یا پکیج

Dependencies :

مهم ترین فیلد package.json هستش که شامل تمام پکیج هاییه که برای پروژمون لازمه و هم چنین ورژن اونارو هم درون خودش داره و هر بار که یک چیزی رو نصب میکنیم داخل این فیلد(یا فیلد های مشابهش) نوشته میشه.

devDependencies :

فیلد devDependencies هم وجود داره که مختص زمان خاصی اند مثلا ممکنه یک سری از نیازمندی هامون فقط برای فاز توسعه باشه مثل prettier

prettier :

فقط برای فرمت دهی کد ها استفاده میشه این نیاز مندی هارو بهتره در devDependencies قرار بدیم .  
 Peer Dependencies:  
یک رابطه بین نسخه های مختلف پلاگین ها و Host Package هاست... یعنی یکسری وابستگی ها وجود داره که نرم افزارمون با یک نسخه ی خاصی کار میکنه و در صورت تغییرات در نسخه خطا دریافت میکنیم اون موقع میتونیم از رابطه Peer استفاده کنیم.

سوال دوم : دلیل حضور فایل قفل (package-lock.json) چیه و چرا باید وجود داشته باشه. محتوای این فایل شامل چیه؟

جواب سوال 2-4 :

هر بار که npm فایل package.json یا ساختار درخت نیازمندی‌های node\_modules را تغییر می‌دهد، بصورت خودکار یک فایل به نام package-lock.json  ایجاد می‌شه .این فایل لیست تمام پکیج ‌های نیازمندی یک پروژه را(چه نیازمندی مستقیم چه غیرمستقیم) به همراه شماره دقیق نسخه‌اش( exact version) در خودش داره .

به ما امکان تولید درخت نیازمندی‌های یکسان را می‌ده در واقع با وجود این فایل براحتی می‌توانیم با diff گرفتن از حالت‌های این فایل در زمان‌های مختلف، تفاوت‌ها و تغییرات درخت نیازمندی‌ها را بصورت واضح ببینیم.

سوال سوم : پوشه node\_modules چیه و چه محتوایی داره؟

جواب سوال 3-4 :

**پوشه node\_modules :**  این پوشه حاوی پکیج هایی است که توسط npm نصب شده است. این پکیج ها ممکن است بواسطه create-react-app نصب شده است و برخی از آنها در فایل package.json فراخوانی شده اند.

سوال چهارم : آیا لازمه پوشه node\_modules رو کامیت و به گیت اضافه کنیم؟ اگه آره چرا، اگه نه چرا؟ در مورد فایل های package.json و package-lock.json چطور؟

جواب سوال 4-4 :

: node modules از روی package Json محاسبه میشه و ساخته میشه و نیازی نیست به گیت فرستاده شود.

اگر فقط package json و package-lock رو داشته باشیم با زدن npm install فایل node modules را میسازد.

ما به گیت فایل هایی رو میفرستیم که امکان ساختن آنها از روی config پروژه فایل پروژه نباشد.

فایل package-lock.json باید به ریپازیتوری گیت کامیت شود تا افراد دیگر بتوانند آن را واکنشی کنند.

سوال پنجم : تحقیق کنید که چطور ممکنه کدهایی که به ظاهر در زبان جاوااسکریپت معتبر نیستن، به عنوان مثال، return کردن تعدادی کد که ظاهرا قیافه تگ های html دارن، در مرورگر میتونن اجرا بشن؟ همون کد رو اگر ما دستی بنویسیم و بخوایم مرورگر اجرا کنه، خطا میخوریم. ولی در پروژه ری اکتی مون، بدون مشکل در مرورگر نشون داده میشن؟

جواب سوال 5-4 :

React.js از یک پایگاه داده به نام Virtual DOM استفاده می کند، که ایجاد یک نسخه مجازی از DOM (Document Object Model) است. این Virtual DOM به روشی است که به شما امکان می دهد، تغییراتی را که به DOM اعمال می کنید، را به صورت مجزا ذخیره کنید، و به طور مداوم، با DOM واقعی مقایسه کنید، اینطوری که فقط از تغییرات نیازی روی DOM واقعی اجرا میشود.

زمانی که شما کد هایی که ظاهرا HTML نیستن را در React قرار می دهید، React هیچ کاری نخواهد کرد، بلکه تنها این کد ها را به عنوان یک متن به توابعی مانند createElement ارسال می کند، و بعدا توسط تابع createElement ، به صورت یک عنصر HTML، در DOM نشان داده می شود.

به همین دلیل، اگر بخواهید کد هایی که ظاهرا HTML نیستن را به صورت دستی در مرورگر نشان دهید، خطایی رخ می دهد، زیرا مرورگر نمی تواند به طور مناسب با این کد ها پردازش کند.