Theory Assignment:

Q① what is 'Npm'?

Ans: Define: Npm is a Javascript package Manager and Behind the Scenes NpM Manages Node packages but does not Stand for Node package Manager.

* npm is the largest Software library (Registry) & also a Software package Manager & Installer.
* NpM is a Standard Repository for all the packages.
* All the packages, libraries & utilities are hosted there & NpM Basically manages them for us.
* All NpM packages are defined in files called package.json.

② what is 'parcel/webpack'? why do we need it?

Ans: 'parcel/webpack' are Bundler's mostly used for Javascript or Typescript. Bundlers are development tools that combine many HTML, CSS & JS Files into a single one that is production ready & loadable in the Browser.

* The main purpose of bundlers is to make it easier to manage dependencies & optimize the performance of web applications by reducing the no. of requests made to the Server.
* Bundler's like parcel uses Minification techniques to improve request load time. Bundling & minification improves load time by reducing the no. of requests to the Server & reducing the size of requested assets (such as CSS & JS).
* Bundler make sure that Code is Minified, cached, compressed, cleaned, optimised before Sending to production.

③ what is '.parcel-cache' Folder ?

Ans: parcel creates '.parcel-cache' for storing the information about the project to reduce the time when we Re-Build our project.
⇒ npx parcel index.html (creates .parcel-cache)

④ what is 'dist' Folder?

Ans: After run the 'build' Command (npx parcel 'build' index.html) in an application the 'dist' folder is generated. The folder contains the build scripts of the application compressed & minified code.

⇒ npx parcel index.html ⇒ creates|Generates a dist folder that contains Development build.

⇒ npx parcel build index.html ⇒ " " " production build.

⑤ what is 'npx' ?

Ans: NPX is a tool for Executing Node packages.

NPM → Manages Node packages But Not easy to execute packages with NPM.

npx parcel index.html ⇒ Execute parcel with entry point being index.html.

npx install parcel ⇒ Installing parcel package.

⑥ what is difference 'Dependencies' vs 'devDependencies' ?

'Dependencies': packages required by our application in both Development & production.

'devDependencies': Packages that are only needed for local development & Testing.

⑦ what is 'Tree Shaking'?

Ans: In 'Tree shaking', It removes unwanted/unused code suppose you have library it has 10 functions we just want to use 2 so, we used 2 & then all the 8 are ignored by parcel, It is known as 'Tree shaking'.

⑧ what is 'Hot Module Replacement'?

Ans: 'HMR' is feature which is enabled by parcel when we use parcel bundler. Its automatically updates the code without fully Refreshing the page in our browser when we do some changes in our code. Because it uses a "File watcher Algorithms" to keep track of file changes in development mode & render those changes on the webpage.

⑨ List Down your Favourite 5 superpowers of parcel & describe any 3 of them in your own words?

Ans: ① zero config
② Development Build
③ create local Dev server (npx parcel index.html)
④ Hot Module Replacement (HMR)
⑤ File watching Algorithm (written in C++)
⑥ Reliable Caching (Faster builds)
⑦ Tree shaking
⑧ Diagnostics
⑨ Content hashing
⑩ Differential Bundling
⑪ Code Splitting.
⑫ Image optimization
⑬ Minification of File (In production code)
⑭ File Bundling
⑮ File compressing

① Zero Config: No configuration and requirement needed for first time usage.

② Development Build:
It gives us a Development build code which can run on Development local server.
npx parcel index.html

③ Dev ~~server~~ local server:
parcel includes a Development server out of the box.
Just run npx parcel index.html ⇒ localhost:1234

④ Diagnostics:
If you make an error in your code or configuration, parcel displays beautiful diagnostics in your terminal & in the browser.
Every error includes a syntax highlighted code frame pointing to exact location where the error occurred, along with hints about how to fix the issue.

⑤ Content Hashing:
parcel automatically includes content hashes in the names of all output files. This enables long-term caching, because the output is guaranteed not to change unless the name does.

Scanned with CamScanner

(10) what is '.gitignore'? what should we add and not add into it?

Ans: '.gitignore' file is used to ignore some files and folders that we write inside the '.gitignore' file.

* If we write files & folders name inside '.gitignore' file then those files & folders are ignored while pushing code to git Repository.

→ We should add Files & Folders that can be Re-generated by npm & server, like node-modules, .parcel-cache, dist etc

→ we should never add those Files & Folders that we can't exact Re-generate through npm & server like package.json, package-lock.json, our html, css & Js files

(11) what is the difference between 'package.json' & 'package-lock.json'?

Ans: package.json:

* It is a configuration file for NPM.
* It keeps track of project Information, like entry point, description and testing etc.
* It keeps track of Approximate/compatible versions of packages.
* In package.json we can write/Define the Bundler versions and Browserslists etc many more things.

~~package.json~~

package-lock.json:

* It contains Detailed Information & Exact version of the packages/dependencies & dependencies depending on dependencies (Transitive dependency), like when install parcel in
* package-lock.json, we have Information & version details about parcel/compressor, parcel/optimizer, parcel/packager, parcel/transformer etc.

* It contains a hash (sha512), which make sure that whatever is on my Local Machine, It is the same version that can be deployed on the production.
So that code will ~~work~~ work both in development & production. Hash (sha512) make sure that.

(12) why should I not modify 'package-lock.json'?

Ans: we should Never Modify package-lock.json file. Because It contains the Detailed Information & Exact version of the packages/Dependencies & Transitive Dependencies.

→ when we change or Modify 'package-lock.json' our code will Break. Because we have to the Exact versions of package that we are consuming in our project.

(13) what is 'node_modules'? It is a good idea to push that on git?

Ans: 'node_modules' Folder is like a database for all the Dependencies or package we installed.

* node_modules folder contains all the code files all the dependencies we installed through NPM.

→ No, it is Not a Good idea to push ~~to~~ 'node_modules' folder on git because it's a very large and heavy Folder. & it can be Easily Re-Generated whenever we want to.

(14) what is Browserslist?

Ans: Browserslists is a tool or package, which can specify the compatibility of the application with the Browser.
we can Manually Modify that Browserslist ~~into~~ package.json

Ex:
```
"browserslist": [
    "last 2 versions",
    "last 2 chrome version"
]
```
} It means that it definitely works on 'last 2 versions' properly but it also work on other version but Not Guaranteed.

⑮ Difference B/w tilde(~) & caret (^) ?

Ans: tilde(~) : "Approximately equivalent to version", will update you to all future ~~patch~~ Major|patch versions, without incrementing minor version.

~~1.2.3 will use releases from 1.2.3 to < 1.3.0~~

* when we have Major updates then tilde (~) will automatically updates.

caret(^) : " compatible with version", will update you to all future Minor|patch versions. without Incrementing the major versions.

* Always suggested to use caret(^), Because it update you to Minor version, which will not break our code.