**Question1**

It’s a typical modern looking application, nothing much, Simply, after looking at this app, my mind started dividing the whole app into different reusable components, like that’s how we developers works,

If you are asking for project setup, then that’s a different story, for spa its different for ssr app its different.

If you are asking that how to efficiently display a large list may be 100’s 1000’s items. Then we use concept called windowing with react-virtualized or react-window. *In simple terms:* This means we don't need to pay the cost of thousands of rows of data being rendered at once.

If you are asking for backend, Then I have to spend much more time with website to understand it completely, only then I can say that we need these many collections/tables in database, and connecting them together to work with them as one unit as needed. Other things like authentication, notification, mailing, msg, etc many solutions available, depends on what we are building.

**Question 2**

React is awesome it gives me the power of react and native JavaScript to build components. A strong hand in react component based architecture and JavaScript can produce a really scalable products.

React is simply the market leader at present, holding around 70-80% market share. So no second thought about my choice.

**Pros and cons of Angular**

Angular is a superheroic JavaScript MVVM framework, founded in 2009, which is awesome for building highly interactive web applications.

**Benefits of Angular:**

* Angular’s created to be used alongside with Typescript. And has exceptional support for it.
* Angular-language-service — which allows intelligence and autocomplete inside of component external HTML template files.
* New features like a generation of Angular based npm libraries from CLI, generation, and development of WebComponents based on Angular.
* Detailed documentation that allows getting the all necessary information for the individual developer without asking his colleagues. However, this requires more time for education.
* One-way data binding that enables singular behaviour for the app which minimized risks of possible errors.
* MVVM (Model-View-ViewModel) that allows developers to work separately on the same app section using the same set of data.
* Dependency injection of the features related to the components with modules and modularity in general.
* Structure and architecture specifically created for great project scalability

**Drawbacks of Angular:**

* Variety of different structures(Injectables, Components, Pipes, Modules etc.) makes it a bit harder to learn in comparison with React and Vue.js, which have an only “Component” in mind.
* Relatively slower performance, according to different benchmarks. On the other hand, it can be easily tackled by utilizing so-called “ChangeDetectionStrategy”, which helps to control the rendering process of components manually.

Companies that use Angular: Companies that use Angular: Microsoft, Autodesk, MacDonald’s, UPS, Cisco Solution Partner Program, AT&T, Apple, Adobe, GoPro, ProtonMail, Clarity Design System, Upwork, Freelancer, Udemy, YouTube, Paypal, Nike, Google, Telegram, Weather, iStockphoto, AWS, Crunchbase.

**Pros and cons of React**

React is a JavaScript library, open sourced by Facebook in 2013, which is great for building modern single-page applications of any size and scale.

**Benefits of React:**

* Easy to learn, thanks to its simple design, use of JSX (an HTML-like syntax) for templating, and highly detailed documentation.
* Developers spend more time writing modern JavaScript, and less time worrying about the framework-specific code.
* Extremely fast, courtesy of React’s Virtual DOM implementation and various rendering optimizations.
* Great support for server-side rendering, making it a powerful framework for content-focused applications.
* First-class Progressive Web App (PWA) support, thanks to the `create-react-app` application generator.
* Data-binding is one-way, meaning less unwanted side effects.
* Redux, the most popular framework for managing application state in React, is easy to learn and master.
* React implements Functional Programming (FP) concepts, creating easy-to-test and highly reusable code.
* Applications can be made type-safe with either Microsoft’s TypeScript or Facebook’s Flow, with both featuring native support for JSX.
* Migrating between versions is generally very easy, with Facebook providing “codemods” to automate much of the process.
* Skills learned in React can be applied (often directly) to React Native development.

**Drawbacks of React:**

* React is unopinionated and leaves developers to make choices about the best way to develop. This can be tackled by strong project leadership and good processes.
* The community is divided on the best way to write CSS in React, split between traditional stylesheets (CSS Modules) and CSS-in-JS (i.e. Emotion and Styled Components).
* React is moving away from class-based components, which may be a barrier for developers more comfortable with Object Oriented Programming (OOP).
* Mixing templating with logic (JSX) can be confusing for some developers at first.

Companies that use React: Facebook, Instagram, Netflix, New York Times, Yahoo, Khan Academy, Whatsapp, Codecademy, Dropbox, Airbnb, Asana, Atlassian, Intercom, Microsoft, Slack, Storybook, and many more.

**Pros and cons of Vue.js**

Vue.js is a JavaScript framework, launched in 2013, which perfectly fits for creating highly adaptable user interfaces and sophisticated Single-page applications.

**Benefits of Vue.js:**

* Empowered HTML. This means that Vue.js has many similar characteristics with Angular and this can help to optimize HTML blocks handling with the use of different components.
* Detailed documentation. Vue.js has very circumstantial documentation which can fasten learning curve for developers and save a lot of time to develop an app using only the basic knowledge of HTML and JavaScript.
* Adaptability. It provides a rapid switching period from other frameworks to Vue.js because of the similarity with Angular and React in terms of design and architecture.
* Awesome integration. Vue.js can be used for both building single-page applications and more difficult web interfaces of apps. The main thing is that smaller interactive parts can be easily integrated into the existing infrastructure with no negative effect on the entire system.
* Large scaling. Vue.js can help to develop pretty large reusable templates that can be made with no extra time allocated for that according to its simple structure.
* Tiny size. Vue.js can weight around 20KB keeping its speed and flexibility that allows reaching much better performance in comparison to other frameworks.

**Drawbacks of Vue.js:**

* Lack of resources. Vue.js still has a pretty small market share in comparison with React or Angular, which means that knowledge sharing in this framework is still in the beginning phase.
* Risk of over flexibility. Sometimes, Vue.js might have issues while integrating into huge projects and there is still no experience with possible solutions, but they will definitely come soon.

Companies that use Vue.js: Xiaomi, Alibaba, WizzAir, EuroNews, Grammarly, Gitlab and Laracasts, Adobe, Behance, Codeship, Reuters.

**Question 3**

Ref is an option, not required thing,

Ref directly gives access to the node. We can do that in many ways.

We can also use ref for component itself. Then we get the access to component instance itself.

We can also pass ref from parent to child component called ref forwarding.

If the ref points to a standard component (DOM node, such as input, select, div etc) then to retrieve the element; you just need to call this.refs.current().

If the ref points to a composite component (a custom component you have created yourself) you need to use the new ReactDOM module like so ReactDOM.findDOMNode(this.refs.ref)

If we declare in-line ref’s then it hinders the render function optimization. As it have to create 1-2 more functions before rendering complete.

It hinders in optimized working of Babel inline plugin.

And the react documents suggest, don’t overuse it,

**Question 4**

It depends on whether we are using cra or our own setup.

But the main goal for production are

Just optimize/minimize everything: js, css, html, images or other assets if we want.

And we should also remove source map while going for production.

And env variables they will be set as we set them.

**Question 5**

SyntheticEvent, are cross-browser wrapper around the browser’s native event. It has the same interface as the browser’s native event, including stopPropagation() and preventDefault(), except the events work identically across all browsers.

The SyntheticEvent is pooled. This means that the SyntheticEvent object will be reused and all properties will be nullified after the event callback has been invoked. This is for performance reasons. As such, you cannot access the event in an asynchronous way.

**Question 6**

A mixin is a class that contains methods for use by other classes without having to be the parent class of those other classes.

They kind of helps in extending multiple methods to classes as in js we can extend to only one class, multiple prototypal inheritance is not supported by js, as done in other scripting language like, Python,

// mixin

let sayHiMixin = {

sayHi() {

alert(`Hello ${this.name}`);

},

sayBye() {

alert(`Bye ${this.name}`);

}

};

// usage:

class User {

constructor(name) {

this.name = name;

}

}

// copy the methods

Object.assign(User.prototype, sayHiMixin);

// now User can say hi

new User("Dude").sayHi(); // Hello Dude!

class User extends Person {

// ...

}

Object.assign(User.prototype, sayHiMixin);

**Question 7**

in JavaScript there is a subtle mechanic called **implicit conversion**, kindly offered by [JavaScript engines](https://www.valentinog.com/blog/js-execution-context-call-stack/). The language does not prevent us from adding numbers and strings:

the **addition operator** + in JavaScript **automatically converts any of the two operands to a string if at least one of them is… a string**!

So Almost every JavaScript entity has a method called toString() which is some ways is a courtesy of Object.prototype.toString.

Also Array.prototype.toString overwrites Object.toString() (also called method shadowing).

'hello' + [89, 150.156, 'mike']

"hello89,150.156,mike"

'hello' + 89

"hello89"