MAD-2 Viva

Vue.js

1. *What is Vue?*
   Vue.js is an open-source JavaScript framework for building user interfaces and single-page applications. Created by Evan You, it focuses on the view layer of applications. Vue extends standard web technologies HTML, CSS, and JavaScript) with an intuitive API that enhances their functionality. It's designed to be incrementally adoptable, allowing developers to integrate it into projects gradually rather than all at once.
2. *Why did you use Vue over React/Jinja2/HTML?*
   Cuz vue is easier to learn with simpler syntax. It has powerful data binding and reactivity out of the box, with rich features alongwith easy integration with existing projects. It aslo offers dynamic, interactive UIs unlike jinja/html which're static.
3. *Did you use Vue CDN or CLI? Why CLI over CDN?*
   - CLI requires installation via npm and a build process, while CDN can be setup using a <script> tag.
   - CLI is best for large, complex apps with modularity & custom builds. CDN is ideal for small projects or adding Vue to existing static pages.
   - CLI supports Single File Components (SFCs), TypeScript and adv tools. CDN is limited to basic Vue functionalities
   - CLI is highly customizable with plugins and configs. CDN has minimal customization options.
   - Use Vue CLI when scalability, maintainability, and advanced tooling are priorities.
   - Use Vue CDN when simplicity and speed of setup are more important than advanced features.
4. *What are Vue lifecycle hooks? Which ones did you use?*
   Vue lifecycle hooks are special methods that allow code execution at specific stages of a component's lifecycle:
   - beforeCreate: Called before the instance is created
   - created: Called after the instance is created but before mounting
   - beforeMount: Called right before mounting the DOM mounted: Called after the DOM has been mounted
   - beforeUpdate: Called when data changes, before the DOM is re-rendered
   - updated: Called after a data change causes the DOM to be re-rendered
   - beforeUnmount: Called before a component is unmounted
   - unmounted: Called after a component has been unmounted

5. *Explain how watch works in Vue.*

   Watch in Vue is used to observe and react to changes in a specific data property or computed value.

   - When the watched value changes, the associated callback function runs automatically.
   - Useful for running asynchronous or expensive operations in response to data changes, such as API calls, form validation, or reacting to route changes.

6. *What is v-cloak?*

   The v-cloak directive is used to hide uncompiled template content until Vue has finished compiling. It prevents users from seeing raw template syntax (like {{message}}) during page loading. It works by applying CSS rules that hide content with the v-cloak attribute until compilation is complete. This is particularly useful for Vue applications that compile in the browser rather than Single File Components.

7. *Difference between v-if and v-show.*

   - v-if: A conditional directive that completely adds or removes elements from the DOM based on a boolean expression. If the expression evaluates to false, the element won't be rendered at all. This is more efficient for elements that rarely change.
   - v-show: Controls element visibility using CSS's display property (display:none). The element remains in the DOM regardless of the condition but is hidden when the expression is false. This is better for elements whose visibility changes frequently.

8. *What are slots in Vue? Did you use them?*

   Slots in Vue are placeholders in a component that allow you to pass and display custom content from the parent component.

   - They make components reusable and flexible by letting parents insert their own content.
   - There are default slots, named slots, and scoped slots for more advanced use cases.

9. *What are props in Vue?*

   Props are custom attributes for passing data from parent components to child components. They establish a one-way data flow from parent to child, preventing child components from directly modifying parent data. Props can be defined with validation requirements including type checking, required flags, default values, and custom validators.

10. *How does Vue Router work?*

    Vue Router is the official routing library for Vue.js. It enables navigation between different views or components in a single-page application (SPA) without reloading the page.

    - It maps URLs to components using routes.
    - Changing the URL updates the displayed component dynamically.
    - Supports features like nested routes, dynamic segments, route guards, and lazy loading.

11. *Why didn’t you use Vuex?*

    - The project is small/simple, so a complex state management library isn’t needed.
    - Vue’s built-in reactivity and props/emit are enough for managing state.
    - Using Vuex would add unnecessary complexity and boilerplate for simple data sharing.

12. *How to implement template inheritance (like Navbar on every page) in Vue?*
    Template inheritance (like having a Navbar on every page) in Vue is implemented using parent and child components.
    How to do it:
    - Create a parent/root component (e.g., App.vue) that includes the Navbar component at the top.
    - Use <router-view> in the parent component to render different page components below the Navbar.

```
    <template>
  <div>
    <Navbar />
    <router-view />
  </div>
</template>
```

13. *What is a for directive in Vue?*
    The v-for directive in Vue is used to render a list of items by iterating over an array or object.
    - Syntax: v-for="item in items"
    - Purpose: Dynamically generates DOM elements for each item in a collection.
14. *What is a filter in Vue?*
    A filter in Vue is a function used to format or transform data before displaying it in the template. A filter formats output for display in Vue templates, improving readability and presentation. (Note: Filters are removed in Vue 3; use computed properties or methods instead.)
15. *Is your project responsive? How did you make it responsive?*
    Yes. table-responsive and using JS and Vue.
16. *What styling did you use (CSS/Bootstrap)?*
    I've used both normal CSS and bootstrap classes and models.
17. *Explain a component and its fetch request with lifecycle hooks.*
    A component in Vue is a reusable, self-contained unit with its own template, logic, and styling. Components often fetch data from APIs when they are created or mounted.
    - created(): Data is fetched before the component is added to the DOM.
    - mounted(): Data is fetched after the component is rendered in the DOM.

Backend & Flask

18. *What is Flask SQLAlchemy?*
    Flask SQLAlchemy is an extension for Flask that adds support for SQLAlchemy, a powerful

Object Relational Mapper (ORM) for Python. Flask SQLAlchemy lets you work with databases in Flask apps using Pythonic syntax and ORM features, making development faster and more efficient.

19. *How to change the database from SQLite to another (required code changes)?*
    1. Update the Database URI in Flask Configuration.
    2. Install Required Database Drivers (PostgreSQL/MySQL).
    3. Update Model Definitions (if needed).
    4. Recreate the DB (Delete old SQLite files and regenerate tables/migrations).
    5. Adjust security/performance settings.
20. *What is ORM?*

    ORM stands for Object Relational Mapper.
    - It is a programming technique that allows you to interact with a database using object-oriented code (classes and objects) instead of writing raw SQL queries.
    - The ORM translates your Python (or other language) objects and methods into SQL commands behind the scenes.
21. *How do you handle authentication in Flask?*

    By using Flask-Login and session management.
22. *What is login_required decorator? Explain its implementation.*

    The login_required decorator is used in web frameworks like Flask to restrict access to certain routes or views, ensuring that only authenticated (logged-in) users can access them. If a user is not logged in, they are redirected to the login page. It wraps around route functions and checks the user's authentication status before allowing access.
23. *How is RBAC (Role-Based Access Control) implemented?*

    Before letting access to anything, we're checking which class does the person belong to? If he's an admin, we open admin dash for him and let him manage everything. While is he's a professional, we let him access his dash and only interact with services requested from him and so on.
24. *Difference between authentication and authorization.*

    Authentication is who the user is? Is he a legit user? What class he belongs to? (Admin, professional, or customer). Authorization is what the user can do? Like customer can access only customer dashboard and can edit and delete his own service requests and so on.
25. *What is CORS?*

    CORS (Cross-Origin Resource Sharing) is a security feature in web browsers that controls how web pages can request resources from a different domain (origin) than the one that served the web page.
26. *What is CSRF? How is it handled?*

    CSRF (Cross-Site Request Forgery) is a web security vulnerability where an attacker tricks a user’s browser into making unwanted requests to a different site where the user is authenticated, potentially performing actions without the user's consent. How handled:

1. CSRF Tokens:
    - The server generates a unique, secret token and includes it in each form or request.
    - The client must send this token back with requests (usually in a hidden form field or custom header).
    - The server verifies the token; if it's missing or invalid, the request is rejected.
2. SameSite Cookies:
    - Set cookies with the SameSite attribute to restrict them from being sent with cross-site requests.
3. User Authentication Checks:
    - Ensure sensitive actions require the user to be authenticated

27. *How are tokens managed (JWT/OAuth)?*
    We've used JWT. JWTs are generated by the server, stored securely on the client, sent with each request, and verified by the server for authentication, ensuring secure and stateless user sessions.
28. *Is Flask open-source? Can it be used for large applications?*
    Yes Flask is open-source. Yes, Flask can be used for large applications by following good design practices (like using Blueprints for modular code). However, for very complex or enterprise-level projects, frameworks like Django may offer more built-in features out of the box.
29. *What is lazy loading?*
    Lazy loading is a design pattern that defers the loading of resources until they're actually needed. In web development, this typically refers to:
    Images that load only when they scroll into view
    JavaScript modules that are imported only when required
    Components that render only when they become visible
30. *How to index in SQLAlchemy?*
    In SQLAlchemy, you can add an index to a database column by using the index=True parameter in your model definition, or by explicitly defining an Index.

Celery & Redis

31. *What is Celery? Explain its components.*
    Celery is an asynchronous task queue/job queue based on distributed message passing. It's focused on real-time operations but also supports scheduling. Celery is used to execute tasks
    outside the HTTP request-response cycle, such as Sending emails, generating reports, periodic tasks etc. Components:
    - Workers: Processes that execute tasks
    - Brokers: Message queues (like Redis or RabbitMQ) that store tasks until a worker can process them

- Result Backends: Stores task results (optional)
- Beat: Scheduler for periodic tasks
- Flower: Web-based monitoring tool for Celery clusters

32. *What Celery jobs have you implemented?*
    Export, daily reminder, monthly reports.
33. *Explain Celery Beat and scheduled tasks.*
    Celery Beat is a scheduler that works with Celery, a Python task queue, to execute tasks automatically at scheduled intervals (like a cron job).
    Scheduled tasks (also called periodic tasks) are functions or jobs you want to run automatically based on a schedule (e.g., every minute, hourly, daily).
34. *How to shut down a Celery task without using the terminal?*
    You can shut down or revoke a Celery task without using the terminal by calling the revoke() method from your application code or an admin interface. We can use Flower dashboard also if it's set up.
35. *What is Redis? Why use it over a normal database?*
    Redis is an open-source, in-memory data store used mainly for caching, real-time data, and fast operations. Why use it over a normal database?
    - Speed: Stores data in RAM, making read/write operations extremely fast.
    - Use Cases: Ideal for caching, session management, leaderboards, and pub/sub messaging.
    - Simplicity: Simple data structures and commands.
36. *Difference between Memcached and Redis.*
    Memcached is a simple, high-speed, in-memory cache.
    Redis is more powerful, offering persistence, advanced data structures, and extra features beyond basic caching.
37. *What is caching? Where did you implement it?*
    Caching is a technique that stores copies of frequently accessed data in a high-speed data storage layer to reduce data retrieval time. In web applications, caching:
    Improves application performance by retrieving data from fast memory instead of slower disk-based storage
    Reduces database load by serving repeated requests from cache
    Provides predictable performance during traffic spikes
    Eliminates database hotspots by caching frequently accessed data
    Increases read throughput IOPS) significantly
38. *What is publish-subscribe in Redis?*
    Publish-Subscribe (pub/sub) in Redis is a messaging pattern where:
    - Publishers send (publish) messages to channels.
    - Subscribers listen (subscribe) to channels and receive messages in real-time when something is published.

Redis pub/sub enables instant, event-driven communication between different parts of an application.

Security

39. *What is a rainbow table?*
    A rainbow table is a precomputed table of reversed password hashes used to crack passwords in a database. It's a time-memory tradeoff technique that allows attackers to recover passwords from hashed values without having to compute the hash for every possible password. Rainbow tables are named for the different "colors" (hash and reduction functions) used in their creation. They're particularly effective against unsalted password hashes.
40. *What was the default hashing algorithm in MAD1?*
    SHA-256 I guess (I am not sure).
41. *What is SHA-256? Can it be decrypted?*
    SHA256 Secure Hash Algorithm 256-bit) is a cryptographic hash function that generates a fixed-size 256-bit 32-byte) hash value from input data of any size. Key characteristics:
    It's a one-way function, meaning it cannot be "decrypted" back to the original input
    Even a small change in input produces a completely different hash
    It's designed to be collision-resistant (difficult to find two inputs that produce the same hash)
    It's used for data integrity verification, digital signatures, and password storage
42. *How many hashing algorithms do you know?*
    MD-5: Produces a 128-bit (16-byte) hash value (shown as 32 hex characters). Not secure for cryptographic purposes—vulnerable to collisions and brute-force attacks.
    SHA-256: Produces a 256-bit (32-byte) hash value (shown as 64 hex characters). Slower than MD5, but much more resistant to collision and preimage attacks.
43. *How to hide passwords in network payloads?*
    - Never send pws in plain texts.
    - Use HTTPS to encrypt all traffic.
    - Hash pws on the server. (This is done)
    - Don't include pws in URLs, headers or response payloads.
44. *What is SQL injection? How to prevent it?*
    SQL Injection is a security vulnerability where an attacker inserts or manipulates SQL queries through user input, potentially gaining unauthorized access to the database, extracting data, or damaging data. Prevention:
    - Use Parameterized Queries / Prepared Statements.
    - Input Validation.
    - Use ORM (Object-Relational Mapping). (They handle param binding, reducing injection risk)

- Least Privilege Principle (Run database operations with limited user permissions).

45. *What is a man-in-the-middle attack?*

    A Man-in-the-Middle (MITM) attack is a security threat where an attacker secretly intercepts and possibly alters communication between two parties (like a user and a server), without them knowing. Purpose being stealing sensitive info or manipulate msgs.

46. *How can you improve the security of your application?*

    - HTTPS Implementation.
    - Token Security (Token expiration and refresh tokens, secure cookie storage).
    - Password requirements (length, spcl chars, etc).
    - Rate limiting to prevent brute force attacks.
    - XSS prevention using input sanitization.
    - CSRF protection by adding CSRF tokens to requests.

47. *Difference between security and privacy.*

    - Security: Involves protecting data and systems from unauthorized access, breaches, or attacks. It focuses on safeguards (like passwords, encryption, firewalls) to keep information safe.
    - Privacy: Concerns how personal or sensitive information is collected, used, shared, and stored. It focuses on an individual's right to control their own data.

APIs & HTTP Methods

48. *Difference between GET, POST, PUT, and PATCH? Can POST be used instead of PUT?*

    - GET: Fetches data from the server. No data modification; only retrieves.
    - POST: Sends new data to the server to create a resource.
    - PUT: Updates/replaces an existing resource completely with new data.
    - PATCH: Updates part of an existing resource; only sends the changed fields.
      Technically it can be used, but not recommended. POST creates new resources, while PUT is meant for full updates. Using them as intended ensures clarity and proper RESTful API design.

49. *How are API headers/auth tokens sent to the API?*

    API headers and authentication tokens are sent to the API as part of the HTTP request headers. The most common method is using the Authorization header: Authorization: Bearer <token>

Caching & Performance

50. *What is memoization?*

    Memoization: A specific optimization technique that stores results of expensive function calls and returns the cached result when the same inputs occur again. It's typically implemented at the function level and is a form of programmatic caching.

51. *Difference between memoization and caching.*
    Memoization: A specific optimization technique that stores results of expensive function calls and returns the cached result when the same inputs occur again. It's typically implemented at the function level and is a form of programmatic caching.
    Caching: A broader concept that involves storing data temporarily in a faster storage medium. Caching can occur at various levels HTTP, database, application) and isn't necessarily tied to function inputs and outputs.
52. *Where did you implement caching? Why is it needed?*
    Caching is a technique used to store frequently accessed data in a temporary, fast-access storage (like RAM) to reduce latency and improve application performance. It avoids repeatedly fetching or recalculating data from slower storage systems, enhancing speed and efficiency.
53. *Difference between local storage and cookies.*
    - Local storage data persists until manually cleared. Cookies can have expiry dates, often sent with every HTTP request.
    - Local storage is only accessible via JS on the client-side. Cookies are accessible by both client-side JS and server via HTTP headers.
    - Local storage is used for storing large amounts of data not needed by the server. Cookies are useful for storing small data pieces (like session IDs) that need to be sent to the server.

Database & SQL

54. *What is ON DELETE CASCADE?*
    ON DELETE CASCADE is a referential action for foreign key constraints that automatically deletes rows in the child table when the corresponding rows in the parent table are deleted. This maintains referential integrity by ensuring that related records are consistently managed.
55. *Explain SQL joins, subqueries, aggregate functions, GROUP BY, HAVING.*
    - JOINS: Used to combine rows from two or more tables based on a related column. (Inner join, left/right/full outer join)
    - Subqueries: A query nested inside another query (e.g., in SELECT, FROM, WHERE).
    - Agg funcs: Compute a single value from a group of rows (COUNT, SUM, AVG, MIN/MAX).
    - GROUP BY: Groups rows sharing a common value (e.g., department) to apply aggregate functions.
    - HAVING: Filters groups after aggregation (similar to WHERE but for grouped data).

Frontend & JavaScript

56. *What is AJAX?*

    AJAX Asynchronous JavaScript and XML) is a set of web development techniques that allows web applications to send and retrieve data from a server asynchronously without interfering with the display and behavior of the existing page. Despite its name, AJAX can use JSON instead of XML, and it's not a programming language but a combination of:
    XMLHttpRequest object or Fetch API
    JavaScript and DOM manipulation
    HTML and CSS for display

57. *How to access the server without using a browser/URL?*

    Several methods exist to access a server without a browser:
    Command-line tools: curl, wget, httpie API clients: Postman, Insomnia
    WebSockets: For bidirectional communication
    SSH/Terminal access: Direct server access
    Custom applications: Using libraries like requests in Python
    Remote desktop protocols: RDP, VNC, or web-based solutions like Apache Guacamole
    Mobile apps: Native applications that communicate with servers

58. *Why so many Vue files? Difference between components and pages.*

    When using Vue CLI to build modular applications, you naturally end up with many .vue files. Each file represents a self-contained component or page, which helps organize and maintain your code efficiently as your app grows.

    - Components are reusable building blocks (buttons, forms, navbars). Pages represent full views/screens (Home, Abt, Profile)
    - Components are nested inside pages or other components, Pages are mapped to routes using Vue router.
    - Components are many small, focused .vue files. Pages are fewer, each one for a different route/view.

59. *How did you implement the search function?*

    Show the search functionality for anyone and explain.

60. *How to center a login form vertically and horizontally?*

    Wrap your form in a flex container with justify-content: center; and align-items: center;, and set the container's height to 100vh (viewport height) for perfect vertical and horizontal centering.

61. *How to move a button in the navbar to the right?*

    Use a flex container for the navbar and add margin-left: auto; to the button to move it to the right.

62. *What is a Single Page Application (SPA)? Difference between SPA and multi-page.*

    A Single Page Application is a web application that loads a single HTML page and dynamically updates the content as the user interacts with it, without reloading the entire

page. SPAs use AJAX and HTML5 to create fluid user experiences similar to desktop applications.

Project-Specific

63. *What improvements did you make from MAD1 to MAD2?*
    We made the project a full stack web app. We made it more responsive using Vue and JS and more secure using more number of security protocols. Also, we made it more modernized using latest techs like Vue3 for example.
64. *What was the hardest part (frontend/backend)?*
    Backend. Designing so many different routes and APIs.
65. *What other features can be added?*
    Say some of the recommended functionalities from project statement.
66. *What technologies did you use in the app?*
    Vue JS, Html, CSS, Bootstrap, Flask, various flask modules, sqlite etc.
67. *What are your thoughts on scaling the application?*
    - Frontend optimization using lazy loading for performance improvement.
    - Backend optimization using caching at more places.
    - DB Scaling using indexing for frequently queried fields.
    - Some type of API Optimization.