Viva notes by Rohit 😊

Question asked to me during viva:

1. First I had to explain each and everything I implemented in my models and why?
2. Explain database and what is the advantage of SQLite?
3. Explain which libraries you have used and why?
4. Explain MVC in detail.
5. What is ORM and two advantages and disadvantages of ORM?
6. What is version control system(VCS) and tell what is other name used for vcs? And what does it mean by “version”?
   I explained vcs but didn’t know the other name of it🙂
7. Explain PUT and POST method.
8. What is the difference between cell spacing and padding?
9. Go on the internet download an image related to grocery store and set as background image in your home page. You can use internet for help.
10. What does it mean by authorisation and authentication?
11. Which is more crucial frontend or backend validation?
12. If I am a user and visited your website and there is some error I am facing to access your website then how can you help me as a developer?
13. What are different types of storage used in browser?
14. What are cookies?
15. Why https is more secure than http? While answering this question I mentioned SSL certificate then he asked about encryption also and how it helps to make connection more secure.

He asked few more questions I don’t remember now. Overall my viva was good my proctor was level3_25 He was humble and super chill. He said that I have made a good project and I have a good conceptual clarity.😊

Based on my viva I have prepared this. I hope it wil help you.

Q1. What is ORM and what are its advantages and disadvantages?

Ans:

ORM stands for "Object-Relational Mapping." It's a programming technique and a set of libraries or frameworks that allow developers to interact with a relational database using programming objects instead of writing raw SQL queries. ORM tools bridge the gap between the object-oriented programming paradigm and the relational database model.

Advantages of ORM:

- **Abstraction of Database Operations:** ORM abstracts away the low-level details of database operations. Developers can work with higher-level programming constructs like classes and objects, making code easier to read and maintain.
- **Productivity and Rapid Development**: ORM can speed up development by reducing the amount of SQL code that needs to be written. It allows developers to focus more on application logic and less on database interactions.
- **Portability:** ORM frameworks often provide a layer of abstraction that makes it easier to switch between different database systems without changing the application code significantly.

- **Database Agnostic**: ORM can help in writing database-agnostic code. You can switch between different database systems (like MySQL, PostgreSQL, SQLite) by changing the ORM configuration rather than rewriting queries.
- **Model-Driven Development:** ORM encourages the use of models, which are object-oriented representations of database tables. This promotes a clearer mapping between the application's data structures and the database schema.
- **Automatic Query Generation**: ORM tools can automatically generate complex SQL queries, reducing the chance of writing incorrect or inefficient queries.

Disadvantages of ORM:

- **Performance Overhead**: ORM tools might introduce performance overhead, as they often generate complex queries that may not be as optimized as hand-tuned SQL queries.
- **Learning Curve**: Learning to use an ORM effectively can require a significant learning curve, especially for complex scenarios.
- **Limited to Supported Features:** ORM tools might not support all the advanced features of a particular database system. This can be limiting if your application requires complex queries or database-specific optimizations.
- **Complex Mappings:** Mapping between objects and database tables might become complex for non-trivial scenarios, leading to maintenance challenges.
- **Loss of Control:** Developers might lose some control over the precise SQL queries being executed, which can be an issue in cases where performance tuning is crucial.
- **Debugging Complexity:** When things go wrong, debugging issues in an ORM setup can be more complex compared to debugging raw SQL queries.

In summary, ORM can be a powerful tool that simplifies database interaction and boosts productivity, especially for simple to moderately complex applications. However, it might not be the best fit for all projects, especially those that require fine-tuned control over database interactions or involve complex query optimizations.

Q2. What is MVC?

Ans:

MVC, or Model-View-Controller, is a software architectural pattern used in designing applications. It divides the application's logic into three interconnected components:

- Model: Represents the data and business logic of the application. This is where we keep the important stuff, like data and rules. It's like the brain of the program.
- View: Handles the presentation and user interface. It displays data from the model to the user and forwards user interactions back to the controller for processing.
- Controller: Acts as an intermediary between the model and the view. It receives user input from the view, processes it, interacts with the model to update data, and then updates the view accordingly.

Advantages of MVC include clear separation of concerns, making the application easier to maintain, test, and extend. It supports parallel development as different teams can work on different components. However, complex applications can sometimes lead to increased complexity due to the number of interactions between components. Overall, MVC promotes a structured and organized approach to building software systems.

Q3. What is difference between authentication and authorisation?

Ans:

- **Authentication:** Authentication is the process of verifying the identity of a user. It answers the question "Who are you?". This is commonly done through usernames and passwords, biometric data (like fingerprints), smart cards, or other authentication methods.
- **Authorization:** Authorization, on the other hand, comes after authentication. Once a user's identity is confirmed, authorization determines what user is allowed to do within a system or application. It answers the question "What are you allowed to do?" Authorization is about defining permissions and access rights. For example, after logging into an application, authorization decides whether the user can view certain pages, edit specific data, or perform certain actions based on their role or privileges.

In simpler terms:

- Authentication: Confirming someone's identity.
- Authorization: Determining what they're allowed to do once their identity is confirmed.

Q4. what is difference in validation and verification

Validation and verification are two terms commonly used in quality assurance and testing processes, but they refer to distinct concepts:

- **Validation**: validation is about ensuring that what has been built is the right thing. It addresses the question, "Are we building the right product?" In other words, Validation is the process of evaluating a product, system, or component during or at the end of the development process to determine whether it meets specified requirements and expectations.

  Imagine you're building an online e-commerce website. Validation in this context would involve ensuring that the features and functionalities you're developing align with the needs and expectations of your users. For instance, you might validate that the shopping cart accurately adds up prices, that users can easily navigate between product categories, and that the checkout process is intuitive for customers.
- **Verification:** Verification involves checking that the actual code and components of your website match the design and requirements. For instance, you might verify that the code for displaying product images aligns with the design specifications, that the payment gateway integration functions as expected, and that the responsive design adapts well to different screen sizes and devices.
  In software development, verification involves activities like code reviews, static analysis, and unit testing to ensure that the code matches the design and requirements.

Q5. What is git “version control”?

Git is a distributed version control system used to track changes in source code during software development. It helps multiple developers work collaboratively on a project while keeping a history of all changes made over time.
In Git, a "version" typically refers to a snapshot of the entire project at a particular point in time. Each version is represented by a commit, which captures the changes made to the files in the project. Each commit has a unique identifier (a hash) and contains information about who made the changes, when the changes were made, and a reference to the changes themselves.

Q6. "What is Bootstrap, and what are its key features and common uses?"

Bootstrap is like a toolkit for building good-looking and mobile-friendly websites easily. It provides ready-to-use pieces like buttons, forms, and menus, so we don't have to start from scratch. People use Bootstrap to make websites quickly and make sure they look nice on phones, tablets, and computers.

Key Features of Bootstrap:

- ***Responsive Design***: Bootstrap is built with a mobile-first approach, ensuring that websites and applications look and work well on various devices and screen sizes, from smartphones to large desktop monitors.
- ***Grid System:*** Bootstrap includes a flexible and responsive grid system that helps in creating layout structures for different screen sizes. This makes it easier to arrange and align content on a page.
- ***Component Library:*** Bootstrap provides a wide range of UI components like buttons, forms, navigation bars, modals, carousels, and more. These components are styled and ready to use, saving developers time and effort in designing and coding from scratch.
- ***Typography and Styling:*** Bootstrap includes typography styles, color schemes, and other design elements that ensure consistent and visually pleasing styling across your web project.
- ***JavaScript Plugins:*** Bootstrap offers a set of JavaScript plugins that enhance interactivity and functionality. Examples include dropdowns, tooltips, popovers, and more.
- ***Customization:*** While Bootstrap comes with a default theme, developers can customize the framework's appearance and behavior to match their project's unique design requirements.

Q7. What is alternative of CSS?

An alternative to CSS (Cascading Style Sheets) would be *inline styling* or using inline styles directly within HTML elements. While CSS is the standard and recommended way to style web content, inline styling can be used to apply styles directly to specific HTML elements without creating a separate stylesheet. However, it's important to note that using inline styles extensively can make your code less organized and harder to maintain, especially in larger projects. CSS allows for more advanced styling options, such as media queries for responsive design, pseudo-classes for interactive elements, and more.

Follow-up question: Is there any other alternatives apart from inline styling.

Yes, there are alternatives to both inline CSS and traditional external stylesheets. One notable alternative is using CSS preprocessors like Sass (Syntactically Awesome Style Sheets) or LESS (Leaner Style Sheets). These preprocessors introduce advanced features and functionalities that can make your styling workflow more efficient and organized.

Q8. what is the difference between truncate and delete?

In short:
"truncate" removes all data from a table quickly, while "delete" removes specific rows based on conditions. The choice between them depends on the specific requirement and whether you want to remove all data or only specific records.

Detailed :

Truncate:

- Truncate is a database operation that removes all rows from a table, essentially wiping out all the data in the table.
- It is a faster operation compared to deleting rows individually, as it doesn't log individual row deletions and doesn't activate triggers or constraints.
- Truncate operation is often used when you want to remove all data from a table but keep the table structure intact.
- Truncate cannot be rolled back (undone) and generally cannot be used if the table is involved in referential integrity (foreign key) relationships or if there are indexed views referring to it.

Delete:

- Delete is a database operation that removes specific rows from a table based on certain conditions or criteria.
- It is a more flexible operation as you can specify which rows to delete using a WHERE clause.
- Delete operations are slower compared to truncating, especially if you're deleting a large number of rows, and they can trigger cascading actions based on foreign key constraints and triggers.
- Delete operations can be rolled back if the database supports transactions.
- Delete is commonly used when you want to remove specific records from a table while preserving other data.

In summary, "truncate" removes all data from a table quickly, while "delete" removes specific rows based on conditions. The choice between them depends on the specific requirement and whether you want to remove all data or only specific records.

Q9. What are http noun and verbs?

In the context of the HTTP (Hypertext Transfer Protocol), the terms "nouns" and "verbs" are often used metaphorically to refer to resources and methods, respectively. These terms help describe how the HTTP protocol is used for communication between clients (such as web browsers) and servers (web servers hosting websites or web services).

Nouns (Resources): In the context of HTTP, "nouns" refer to the resources or entities that clients want to interact with. A resource can be anything that has a URL (Uniform Resource Locator) and can be identified on the web, such as a webpage, an image, a video, or a specific data item in a web service. Nouns in HTTP are represented by URLs.

Verbs (HTTP Methods): "Verbs" in the context of HTTP are actually the HTTP methods (also known as HTTP verbs) that clients use to perform actions on resources. These methods define the type of operation the client wants to carry out on a resource. The most common HTTP methods are:

- GET: Retrieve data from the server (nouns/resources).
- POST: Send data to the server to create a new resource.
- PUT: Send data to the server to update or replace an existing resource.
- DELETE: Request the removal of a resource from the server.
- PATCH: Send data to the server to partially update a resource.

When a client sends an HTTP request to a server, it typically includes a URL (identifying the noun/resource) and an HTTP method (indicating the verb/action to perform on that resource). This combination of nouns and verbs forms the foundation of the HTTP communication, allowing clients to request and manipulate resources on servers.

Q10. What is a REST API, and what are its core principles, benefits, and common use cases in web development?"

REST API, or Representational State Transfer Application Programming Interface, is a set of rules and conventions for building and interacting with web services REST APIs are commonly used to build web services that enable applications, websites, or different systems to interact with each other. For instance, a mobile app might use a REST API to retrieve user data from a server, or a website might use an API to fetch product information from an online store's database.

Overall, REST APIs provide a standardized and flexible way to create interoperable and scalable web services by adhering to a set of design principles.

Key features and concepts of REST APIs:

- **Resources:** In REST, everything is treated as a resource. A resource can be an object, a piece of data, or anything that can be uniquely identified.

- **HTTP Methods:** REST APIs use HTTP methods (verbs) to perform actions on resources. Common methods include GET (retrieve data), POST (create new data), PUT (update data), and DELETE (remove data).
- **Stateless:** Each API request from a client to the server must contain all the necessary information for the server to understand and fulfill the request. The server doesn't store any information about previous requests.
- **Uniform Interface:** REST APIs provide a consistent and uniform way to interact with resources, using standard HTTP methods and status codes.
- **Representation:** Data is represented in various formats, often including JSON or XML, to ensure compatibility with different clients and platforms.
- **Client-Server Architecture:** The client (requester) and server (provider) are separate entities that communicate over the API. This separation allows for independent development and scalability.
- **Statelessness:** Each request from a client to the server must contain all the information needed for the server to understand and respond. The server doesn't store any client state between requests.

Follow-up question: Is there any difference in rest and restful
In simpler terms, "REST" refers to the architectural style and principles, while "RESTful" describes an application or system that follows those principles. An application that is RESTful adheres to the design guidelines of REST.

Q.11 What is CRUD?

CRUD is an acronym that stands for Create, Read, Update, and Delete. It represents the four basic operations that are commonly used to manage and manipulate data in a database or any other persistent storage system. These operations are fundamental in many software applications for performing various tasks involving data management.

- **Create**: This operation involves adding new data records or entities to a database. It corresponds to the process of inserting new data into the system.
- **Read:** The Read operation is about retrieving and viewing data from the database. It involves querying the database to retrieve specific records or all records that match certain criteria.
- **Update:** The Update operation allows modifying existing data records in the database. It involves making changes to the attributes or values of a particular record.
- **Delete:** This operation is used to remove data records from the database. It involves deleting records that are no longer needed or relevant.

Q.12 What's the difference between "controllers" and "APIs" in software development?

- Controllers are responsible for managing the flow of data and interactions within a single application, often following a specific architectural pattern like MVC.
- APIs provide a standardized way for different software systems to communicate and interact, enabling data exchange and functionality access between different applications or components.

Detailed answer:

Controllers: Controllers are components within the software's architecture that handle the logic of receiving and processing requests from users or other parts of the application. They are commonly associated with the Model-View-Controller (MVC) or similar design patterns. Controllers receive input, usually in the form of user interactions, process that input, interact with the relevant data or business logic, and then produce an appropriate response.

APIs (Application Programming Interfaces): APIs are a set of defined rules and protocols that allow different software components to communicate with each other. APIs provide a way for developers to access the functionality of other software systems or services without needing to understand their internal workings. APIs can be used to retrieve data, send data, or perform specific actions in a remote system. APIs are widely used to enable interaction between different software applications, both within a single application (as part of a modular architecture) and between different applications (for integration purposes).

Q.13 which is the more secure http or https and why?

HTTPS is more secure than HTTP. The key difference between the two lies in the level of security they provide for data transmission over the internet.

HTTP (Hypertext Transfer Protocol): HTTP is a protocol used for transferring data between a web browser and a web server. However, it does not provide any inherent security mechanisms for the data being transmitted. The data is sent in plain text, which means that if someone intercepts the communication, they can easily read the information being exchanged. This lack of encryption makes HTTP connections susceptible to various security threats, including data tampering, and man-in-the-middle attacks.
HTTPS (Hypertext Transfer Protocol Secure): HTTPS, on the other hand, is a secure version of HTTP. It incorporates encryption mechanisms, typically using SSL (Secure Sockets Layer) or its successor, TLS (Transport Layer Security), to establish a secure and encrypted connection between the web browser and the server. This encryption ensures that data sent between the browser and the server is encrypted and cannot be easily intercepted or read by malicious actors.

Key reasons why HTTPS is more secure than HTTP:

- **Data Encryption:** With HTTPS, the data transmitted between the client and server is encrypted, making it extremely difficult for unauthorized parties to decipher the information even if they intercept it.
- **Data Integrity:** HTTPS includes mechanisms to verify that the data being sent has not been altered during transmission. This ensures that the information received by the recipient is the same as what was sent by the sender.
- **Authentication:** HTTPS involves the use of digital certificates, which are issued by trusted Certificate Authorities (CAs). These certificates verify the identity of the website or server, helping users ensure they are connecting to the legitimate site and not a malicious one.
- **Protection Against Man-in-the-Middle Attacks:** The encryption provided by HTTPS prevents attackers from intercepting and modifying the communication between the client and server without being detected.

Q.14 What is VCS?

VCS stands for "Version Control System." It is a software tool used by developers and teams to manage changes to source code and other files over time. VCS helps keep track of different versions of files, allows collaboration among developers, and provides mechanisms to review, merge, and manage code changes. **VCS are sometimes known as SCM (Source Code Management) tools or RCS (Revision Control System).**

Common VCS systems include:
- Git: A distributed VCS widely used for its speed, flexibility, and popularity among open-source projects. Git repositories can be hosted on platforms like GitHub, GitLab, and Bitbucket.
- Subversion (SVN): A centralized VCS that stores a single copy of the repository on a server. It is often used in more traditional version control setups.

- Mercurial: Another distributed VCS similar to Git but with a different underlying model.