Interview Questions-1

(Practice Project)

PW SKILLS

Easy

1. What is a web framework, and why is it important?

Ans: A web framework is a software framework that is designed to support the development of web applications including web services, web resources, and web APIs. Web frameworks automate the overhead associated with common activities performed in web development, allowing developers to focus on the unique aspects of their applications rather than on routine tasks like routing, handling requests, or interacting with databases.

2. How does Flask differ from Django?

Ans: Flask is a microframework, which means it provides the basic features necessary for web development, allowing for flexibility and simplicity. Django is a full-stack framework, offering a lot of built-in features and a "batteries-included" approach, which can make it more suitable for large-scale applications where these features are required.

3. What is Streamlit primarily used for?

Ans: Streamlit is used for building interactive and data-driven web applications, particularly for data science and machine learning projects. It allows developers to create visually appealing dashboards and data visualizations with minimal code.

4. What is the main purpose of Django's ORM?

Ans: Django's Object-Relational Mapping (ORM) system allows developers to interact with the database using Python code rather than writing raw SQL queries. It abstracts the database interactions, making it easier to manage and query data models.

5. Explain the concept of routing in Flask.

Ans: Routing in Flask involves mapping URLs to specific functions or views within a web application. When a user accesses a particular URL, the corresponding view function is executed to handle the request and generate a response.

6. What is Jinja2, and how is it used in Flask?

Ans: Jinja2 is a templating engine for Python, used in Flask to generate HTML dynamically by injecting data from the backend into templates. It allows for creating dynamic web pages with placeholders that are replaced by actual data at runtime.

7. What are the key components of a Django application structure?

Ans:A typical Django application structure includes components like models.py (defining data models), views.py (handling logic and returning responses), urls.py (URL routing), templates (HTML files for rendering), and static (static files like CSS, JavaScript, and images).

Medium

8. Question: Describe the MTV architecture in Django.

Ans: The MTV (Model-Template-View) architecture in Django separates the application into three layers: Models (representing the data structure), Templates (handling the presentation layer), and Views (processing user input and returning responses). This separation ensures a clean and maintainable codebase.

9. What are Flask extensions, and why are they useful?

Flask extensions are add-ons that provide additional functionality to a Flask application, such as database integration, authentication, or form handling. They allow developers to easily extend Flask's capabilities without having to build these features from scratch.

10. How does Django handle authentication and authorization?

Ans: Django provides a built-in authentication system that includes models and views for handling user registration, login, logout, and password management. It also includes middleware and decorators to enforce user permissions and access control across the application.

11. Explain how Streamlit handles state and user interaction.

Streamlit handles state through the use of session state, which allows data to persist across multiple user interactions. It also provides various widgets, such as sliders, buttons, and text inputs, to facilitate interactive user experiences.

12. What is the role of the 'manage.py' file in a Django project?

Ans: The 'manage.py' file in Django is a command-line utility that allows developers to interact with the Django project. It provides commands to run the development server, apply database migrations, create applications, and more.

13. Compare the deployment considerations for Flask, Django, and Streamlit.

Ans: Flask and Django require setting up a web server (like Gunicorn or uWSGI) and configuring it to serve the application, often behind a reverse proxy (like Nginx). Streamlit, on the other hand, can be deployed easily on Streamlit Cloud with minimal configuration but may require more setup for custom deployments on other platforms.

14. How does Django's 'forms' module help in handling user input?

Ans: Django's 'forms' module provides a way to create and manage HTML forms in a Pythonic manner. It handles form validation, processing, and rendering, making it easier to capture and validate user input before saving it to the database or using it in the application logic.

15. Explain the use of Django REST Framework for building APIs.

Ans: Django REST Framework (DRF) is a powerful and flexible toolkit for building Web APIs in Django. It provides features like serialization, authentication, permissions, and viewsets, making it easier to build RESTful APIs that integrate seamlessly with Django's ORM and authentication system.

16. How does Flask support asynchronous operations, and what are its limitations?

Ans: Flask supports asynchronous operations through the use of asynchronous Python libraries like asyncio or by running background tasks using Celery. However, Flask itself is not inherently asynchronous and requires additional setup to handle concurrent requests effectively, especially under heavy loads.

17. Discuss the scalability of Django compared to Flask and Streamlit.

Ans: Django is generally more scalable out of the box due to its full-stack nature, built-in caching, and robust ORM. Flask, being a microframework, requires additional components to handle scalability but allows for more fine-grained control. Streamlit, while excellent for rapid prototyping and data apps, may require more customization and external services to scale effectively.

18. What are Django's middleware, and how do they enhance the framework's functionality?.

Ans: Middleware in Django is a series of hooks or layers that process requests and responses before reaching the view or after leaving it. Middleware can handle tasks such as session management, user authentication, content compression, or modifying request/response data, enhancing the overall functionality and security of the application.

19. Describe how Flask's 'blueprints' help in organizing large applications.

Ans: Flask's 'blueprints' allow developers to organize large applications into smaller, reusable components. Each blueprint can encapsulate routes, templates, and static files related to a specific feature or module, promoting modularity and easier maintenance of the application.

20. How does Streamlit optimize performance for large datasets and complex visualizations?

Ans: Streamlit optimizes performance for large datasets by using data caching and lazy evaluation. It only re-renders components that change, reducing unnecessary computations. For complex visualizations, developers can use efficient libraries like Plotly or Altair, which Streamlit integrates with seamlessly, to ensure smooth rendering.