Question 1:

What type of software is Task2 primarily designed to help developers build?

a. Data analysis platforms (e.g., data dashboard)

b. Web applications and APIs (e.g., personal portfolio website)

c. Machine learning pipelines (e.g., model deployment interface)

d. Desktop GUI applications (e.g., task management app)

e. I don't know

Question 2:

What are the key components of this codebase? (Choose all that apply)

a. JSON Handling
b. Blueprints
c. config
d. Database
e. Middleware
f. HTTP
g. Sansio
h. All of the above
i. I don't know

Question 3:

What is the role of the App component (i.e., location of app.py) in a Flask project? (Choose all that apply)

a. To store all static files used in the project.

b. To define the database schema and relationships for the application.

c. To configure the Flask application, initialize routes, and start the server.

d. To act as the central entry point for the application.

e. To validate request and response objects using middleware.

f. All of the above.

g. I don't know.

Answers and Reasons:

a. To store all static files used in the project: Wrong: Static files are stored in a static directory, not in app.py.

b. To define the database schema and relationships for the application: Wrong: Database schema and relationships are typically defined in models.py, not in app.py.

c. To configure the Flask application, initialize routes, and start the server: Correct: app.py configures the application, defines routes, and uses app.run() to start the server.

d. To act as the central entry point for the application: Correct: app.py is usually the main entry point for a Flask project, combining all components like routing, configuration, and initialization.

e. To validate request and response objects using middleware: Wrong: Middleware can be configured elsewhere, but app.py does not perform direct validation of requests and responses.

f. All of the above: Wrong: Not all options are correct, as options a and b are invalid.

g. I don't know: This option provides no insight into the question.

Question 4:

Which description is correct for the APP core component and its relationship with other components in Flask? (Choose all that apply)

a. The APP core requires developers to manually implement JSON handling since it lacks a built-in provider.
b. Middleware is a required component of the APP core, and applications cannot function without it.
c. The APP core provides the foundation for CLI commands to interact with application configurations.
d. The APP core handles routing, request processing, and response generation.
e. All of the above.
f. I don't know.

Answers and Reasons:

a. The APP core requires developers to manually implement JSON handling since it lacks a built-in provider: Wrong: Flask includes built-in support for JSON handling through the json module and provides tools for parsing and serializing JSON.

b. Middleware is a required component of the APP core, and applications cannot function without it: Wrong: Middleware is optional in Flask. While it enhances functionality, such as adding pre- or post-processing to requests, applications can run without it.

c. The APP core provides the foundation for CLI commands to interact with application configurations: Correct: The APP core in Flask supports the Flask application object, which integrates with CLI commands, allowing developers to run commands such as flask run and flask db migrate.

d. The APP core handles routing, request processing, and response generation: Correct: The APP core processes incoming HTTP requests, matches them to routes, executes view functions, and generates appropriate HTTP responses.

e. All of the above: Wrong: Since options a and b are incorrect, this cannot be the correct answer.

f. I don't know: This option provides no insight into the question.

Question 5:

What is the relationship between the App Module and the Config Module in Flask? (Choose all that apply)

a. The Config Module provides configuration settings that the App Module uses to manage the application's behavior.

b. The App Module automatically loads configuration settings from the Config Module.

c. The Config Module is required to initialize the App Module.

d. The App Module uses the config object to access application-specific settings, such as DEBUG or SECRET_KEY.

e. The Config Module provides environment-based configurations that can be loaded into the App Module.

f. The App Module depends on the Config Module to define all middleware used in the application.

g. All of the above.

h. I don't know.

Answers and Reasons:

a. The Config Module provides configuration settings that the App Module uses to manage the application's behavior: Correct: The Config Module is used to define application settings like DEBUG, DATABASE_URI, and SECRET_KEY, which are critical for the App Module to function.

b. The App Module automatically loads configuration settings from the Config Module: Correct: When the Flask app object is initialized, it automatically loads default and custom configurations from the Config Module via app.config.

c. The Config Module is required to initialize the App Module: Wrong: While the App Module uses configurations, it does not require the Config Module to be explicitly defined unless specific custom configurations are needed.

d. The App Module uses the config object to access application-specific settings, such as DEBUG or SECRET_KEY: Correct: Developers can use app.config['SETTING_NAME'] to access or modify configurations.

e. The Config Module provides environment-based configurations that can be loaded into the App Module: Correct: Flask supports loading configurations based on different environments (e.g., development, testing, production) using the Config Module.

f. The App Module depends on the Config Module to define all middleware used in the application: Wrong: Middleware is defined independently and is not directly dependent on the Config Module.

g. All of the above: Wrong: Since options c and f are incorrect, this cannot be the correct answer.

h. I don't know: This option provides no insight into the question.

Question 6:

What is the role of the Blueprint Module in Flask? (Choose all that apply)

a. To handle HTTP requests and responses directly.

b. To enable modular organization of application functionality, such as routes and templates.

c. To replace the App Module as the central entry point for the Flask application.

d. To allow reusable components to be shared across multiple applications.

e. To enable developers to define and group related routes and views.

f. To enhance security by validating incoming requests.

g. All of the above.

h. I don't know.

Answers and Reasons:

a. To handle HTTP requests and responses directly: Wrong: The Flask App Module handles HTTP requests and responses. The Blueprint Module does not directly process requests or generate responses but organizes related routes and functionality.

b. To enable modular organization of application functionality, such as routes and templates: Correct: Blueprints allow developers to group related functionality, such as routes, templates, and static files, into modular components for better organization.

c. To replace the App Module as the central entry point for the Flask application: Wrong: The App Module remains the central entry point, while the Blueprint Module is used to structure and organize application features.

d. To allow reusable components to be shared across multiple applications: Correct: Blueprints can be designed as reusable components that can be registered with multiple Flask applications.

e. To enable developers to define and group related routes and views: Correct: Blueprints allow developers to define a set of related routes, views, and associated resources, making the codebase more maintainable.

f. To enhance security by validating incoming requests: Wrong: Security is typically handled by Flask's middleware or extensions like Flask-Security, not the Blueprint Module.

g. All of the above: Wrong: Since options a, c, and f are incorrect, this cannot be the correct answer.

h. I don't know: This option provides no insight into the question.

Question 7:

What is the relationship between the App Module and the Blueprint Module in Flask? (Choose all that apply)

a. The App Module is required to define and register a Blueprint Module.
b. The Blueprint Module allows the App Module to organize application functionality into smaller, modular components.
c. The Blueprint Module replaces the App Module as the central entry point for the Flask application.
d. The App Module can register multiple Blueprints to handle different routes and functionality.
e. The App Module depends on the Blueprint Module for basic request and response handling.
f. All of the above.
g. I don't know.

Answers and Reasons:

a. The App Module is required to define and register a Blueprint Module: Correct: A Blueprint must be registered with the Flask App instance to associate its routes, templates, and functionality with the main application.

b. The Blueprint Module allows the App Module to organize application functionality into smaller, modular components: Correct: Blueprints provide a way to break an application into modular components, improving maintainability and scalability.

c. The Blueprint Module replaces the App Module as the central entry point for the Flask application: Wrong: The App Module remains the central entry point; Blueprints only help organize functionality and routes.

d. The App Module can register multiple Blueprints to handle different routes and functionality: Correct: Flask allows multiple Blueprints to be registered with the App Module, enabling better separation of concerns.

e. The App Module depends on the Blueprint Module for basic request and response handling: Wrong: The App Module itself can handle requests and responses directly; the Blueprint Module is an optional feature to improve organization.

f. All of the above: Wrong: Since options c and e are incorrect, this cannot be the correct answer.

g. I don't know: This option provides no insight into the question.

Question 8:

Under the Blueprint Module, what is the purpose of register() in Flask? (Choose all that apply)

a. To add the Blueprint's routes and resources to the main Flask application.
b. To initialize middleware exclusively for the Blueprint.
c. To connect the Blueprint to the App Module, allowing its functionality to be integrated into the application.
d. To register extensions, such as SQLAlchemy or Flask-Migrate, within the Blueprint.
e. To associate static files and templates specific to the Blueprint with the application.
f. All of the above.
g. I don't know.

Answers and Reasons:

a. To add the Blueprint's routes and resources to the main Flask application: Correct: The register() method integrates the Blueprint's routes, views, and other functionality into the main application, making them accessible.

b. To initialize middleware exclusively for the Blueprint: Wrong: Middleware is usually global to the application and not limited to a specific Blueprint. The register() method does not handle middleware initialization.

c. To connect the Blueprint to the App Module, allowing its functionality to be integrated into the application: Correct: The register() method links the Blueprint to the main application, enabling modular functionality to be seamlessly added.

d. To register extensions, such as SQLAlchemy or Flask-Migrate, within the Blueprint: Wrong: Extensions are typically initialized globally within the App Module, not within a Blueprint or during registration.

e. To associate static files and templates specific to the Blueprint with the application: Correct: When a Blueprint is registered, its static files and templates (if specified) become accessible through the main application.

f. All of the above: Wrong: Since options b and d are incorrect, this cannot be the correct answer.

g. I don't know: This option provides no insight into the question.

Question 9:

Under the Blueprint module, what is the purpose of get_send_file_max_age() in Flask? (Choose all that apply)

a. To determine the maximum age (in seconds) for static files before they are cached.

b. To set a global cache-control header for all static files served by the application.

c. To customize the cache duration for static files specific to a Blueprint.

d. To override the default cache duration defined by the App Module.

e. To ensure static files are always reloaded without caching.

f. All of the above.

g. I don’t know.

Answers and Reasons:

a. To determine the maximum age (in seconds) for static files before they are cached: Correct: The get_send_file_max_age() method is called to compute the cache expiration time for static files served by a Blueprint.
b. To set a global cache-control header for all static files served by the application: Wrong: This method applies only to static files associated with a specific Blueprint, not globally.
c. To customize the cache duration for static files specific to a Blueprint: Correct: This method allows Blueprints to define custom caching behavior for their static files independently of the rest of the application.
d. To override the default cache duration defined by the App Module: Correct: If the App Module has a default caching duration, get_send_file_max_age() can override it for static files linked to a particular Blueprint.
e. To ensure static files are always reloaded without caching: Wrong: While setting a very low or zero value for the max age could disable caching, this is not the primary purpose of the method.
f. All of the above: Wrong: Since options b and e are incorrect, this cannot be the correct answer.

g. I don’t know: This option provides no insight into the question.