**Technical Questions**

**Part 1: Code Review and Refactoring (30%)**

**Task 1**

To identify a certain function, I usually use CodeIgniter code in the application in config folder and then the php code to check the base URL which then leads to routes.php file and in there we get the file URL and controller and functions so we can easily find out. But in this existing system, the application folder path with page-based method routing is used so it’s harder to identify the function and to implement new.

**Part 3: Theoretical Knowledge and Best Practices (10%)**

**Task 8**

**PHP 8 features and best practices**

**Features**

* **Union Types:** With the introduction of union types in PHP 8, a variable can now take several value types.
* **Named Arguments:** Code readability is enhanced with named arguments, which let you send values to a function by naming the parameter.
* **Match Expression:** A more potent and adaptable substitute for the switch statement is match.
* **Nullsafe Operator:** To minimize the need for explicit null checks, you can act on an object only if it is not null using the Nullsafe operator (?->).
* **Attributes:** Classes, methods, and properties can all have metadata added to them via attributes.
* **Constructor Property Promotion:** Classes, methods, and properties can all have metadata added to them via attributes.

**PHP best Practices**

* **Maintain Updating Your PHP Version:** Update your PHP version frequently to take advantage of the newest features, security enhancements, and bug fixes.
* **Employ Careful Typing:** Use the declare(strict\_types=1); declaration at the start of your files to enforce strict typing and detect type-related issues early.
* **Types of Leverage Unions:** Use union types to give function and method argument types of greater flexibility.
* **Use Identified Arguments**: When calling functions, especially those with numerous parameters, use named arguments to make the code easier to read.

**Understanding of MVC architecture.**

A software design pattern called MVC (Model-View-Controller) divides an application into three interrelated parts:

**Model (M):**

* Represents the application's data and business logic.
* Maintains rules, logic, and data while guaranteeing data integrity.

**View (V):**

* Represents the application's display and user interface.
* Gives the user information and records their input.

**Controller (C):**

* serves as a middleman, taking in and processing user input before updating the model and view.
* controls the data transfer between the view and the model.

MVC encourages concern separation, which improves the modularity, maintainability, and scalability of applications. By dividing the data manipulation, user interface, and control flow into separate components, it improves code structure.

**Web security principles relevant to PHP applications**

* **Input Validation and Sanitization:** Verify and clean user input to stop harmful information from getting into the program. Make use of filter\_input() and data type validation.
* **Parameterized Statements and Prepared Statements:** To stop SQL injection attacks, utilize prepared statements or parameterized statements to keep SQL code and user input apart.

* **Preventing Cross-Site Scripting (XSS):** Get rid of and clean up output data to stop XSS attacks. When displaying user-generated content, make use of functions such as htmlspecialchars().
* **Cross-Site Request Forgery (CSRF) Protection:** Use anti-CSRF tokens to verify and make sure that requests are authentic, safeguarding against unwanted activities brought about by faked requests.
* **Security of Sessions:** Make use of session\_regenerate\_id(), secure session storage, and configuring secure session parameters, among other secure session handling techniques.

**Strategies for optimizing PHP applications for performance.**

* **Code profiling:** Use profiling tools such as Xdebug to examine code execution times and memory utilization in order to pinpoint bottlenecks.
* **Mechanisms of Caching:** Use opcode caching, such as OPCache, to cache precompiled script bytecode and eliminate the need to parse the script for each request.
* **Make Use of Effective Database Queries:** Optimize database queries by using prepared statements to avoid SQL injection, indexing tables, and retrieving only the data that is required.
* **Indolent loading:** Reduce the time it takes for an application to load by implementing lazy loading for resources and classes.