Web Engineering

Made by Md. Mehedi Hasan Rafy

Year 2020

Section - A

1. a) What is a Web App? Discuss how Web Apps are different from traditional software?

Solution:

Web Application

A web application, often referred to as a web app, is a software application that runs on web servers and is accessed through a web browser over a network, typically the internet.

Web Apps vs Traditional Software

Attribute	Web Apps	Traditional Software
Network Intensiveness	Relies on internet or network connectivity to function.	Often works offline; network is optional.
Concurrency	Supports multiple users accessing simultaneously.	Usually single-user or limited concurrency.
Unpredictable Load	Traffic can vary drastically; must handle spikes.	Load is generally predictable, installed per system.
Performance	Needs optimization for varying network speeds and server load.	Performance mostly depends on local machine resources.
Availability	Must be available 24/7 for users.	Usually only needs to be available when installed/used.
Data Driven	Frequently interacts with databases and real-time data.	May store data locally; interaction with databases is optional.

Content Sensitive	Content can be dynamic and personalized per user.	Content is mostly static and uniform.
Continuous Evolution	Updated regularly without user intervention.	Updates require user action (installation/upgrade).
Immediacy	Accessible instantly via browser; no installation required.	Requires installation before use.
Security	Faces web-specific threats (XSS, CSRF, SQL injection).	Security is mostly local or system-based.
Aesthetics	UI/UX is critical for user engagement and satisfaction.	UI may be less flexible, often depends on OS standards.

b) Briefly discuss various categories of web applications.

Solution:

Document-Centric	***Interactive Web Applications***
Description: Predecessors of modern web apps; static HTML documents stored on a web server and delivered to clients. **Features:** Updated manually, simple, stable, short response time, but can lead to outdated information or inconsistencies. **Examples:** Static homepages, small business websites.	**Description:** Use CGI and HTML forms to generate dynamic pages. **Features:** User interaction, dynamically generated content. **Examples:** Virtual exhibitions, news sites, timetable systems.
Transactional	***Workflow-Based***
Description: Highly interactive, data-driven; allow users to update information efficiently. **Features:** Consistent handling of large content, often tied to databases. **Examples:** Online banking, e-commerce, booking systems.	**Description:** Automate workflows within or between organizations. **Features:** Require structured processes, robust and flexible, handle complex interactions. **Examples:** B2B e-commerce solutions, e-government portals, healthcare workflow

<table>
<tr><td></td><td>systems.</td></tr>
<tr><td>Collaborative</td><td>Social</td></tr>
<tr><td>Description: Support group work and communication.

Features: Shared information, workspaces, high degree of communication.

Examples: Wikis, Google Meet, Google Classroom, scheduling systems.</td><td>Description: Connect users with similar interests and facilitate community interaction.

Features: Identity sharing, interest-based connections.

Examples: Blogs, Facebook, Friendster.</td></tr>
<tr><td>Portal-Oriented</td><td>Ubiquitous</td></tr>
<tr><td>Description: Central hubs providing access to multiple, heterogeneous sources.

Features: Single point of access, specialized information delivery.

Examples: Business portals, healthcare portals, marketplace portals.</td><td>Description: Accessible everywhere, seamlessly integrated into users’ lives across devices.

Features:

Cross-platform compatibility

Responsive design

Offline functionality

Cloud-based infrastructure

Location awareness and personalization

Examples: Mobile banking apps, cloud-based productivity apps.</td></tr>
<tr><td colspan="2">Semantic</td></tr>
<tr><td colspan="2">Description: Present data in a machine-readable form to support knowledge management.

Features: Facilitates content syndication, recommender systems, and more intelligent search.

Examples: Semantic search engines, intelligent recommendation platforms.</td></tr>
</table>

2. a) What do you understand by web engineering?

Solution:

Web Engineering

Web engineering is a discipline that focuses on the systematic, disciplined, and quantifiable approach to the development, operation, and maintenance of high-quality web-based systems and applications.

It involves applying engineering principles, methodologies, and techniques to design, build, deploy, and manage web-based solutions that meet user requirements, adhere to industry standards, and satisfy business objectives.

b) Discuss the generic activities that are the parts of a Web Engineering framework.

Solution:

Generic Web Engineering Framework Activities

1. **Communication**
    - Heavy interaction and collaboration with stakeholders.
    - Focuses on ***requirements gathering*** and related activities.
    - Ensures developers understand what the customer needs.
2. **Planning**
    - Establishes an ***incremental plan*** for the web engineering work.
    - Defines:
        - Web engineering actions and technical tasks
        - Potential risks
        - Required resources
        - Work products to be produced
        - Work schedule and timelines
3. **Modeling**

- Creation of ***models*** to help developers and customers understand:
- Web application requirements
- Design strategies to meet those requirements
- Can include data models, navigation models, or architectural diagrams.

4. **Construction**
   - ***Coding*** the application (HTML, XML, PHP, JavaScript, etc.).
   - ***Testing*** mechanisms to detect and correct errors in the code.
5. **Deployment**
   - Delivering a ***working web application increment*** to the customer.
   - Customer evaluates the increment and provides ***feedback*** for improvements.

c) What tasks are required to develop an increment plan for Web Apps?

Solution:

Tasks Required to Develop an Increment Plan for Web Apps

1. **Refine the Description** – Clarify what the WebApp increment will deliver.
2. **Select the Increment** – Decide which part or functionality will be delivered now.
3. **Estimate Effort and Time** – Calculate the resources, time, and effort needed.
4. **Assess Risks** – Identify possible risks associated with the delivery.
5. **Define Development Schedule** – Establish timelines and milestones.
6. **Establish Work Products** – List deliverables (documents, code, models) for each activity.
7. **Define Change Control Approach** – Plan how changes will be handled during development.
8. **Set Quality Assurance Approach** – Specify how quality will be verified (testing, reviews).

3. a) Discuss the life cycle of a http request.

Solution:

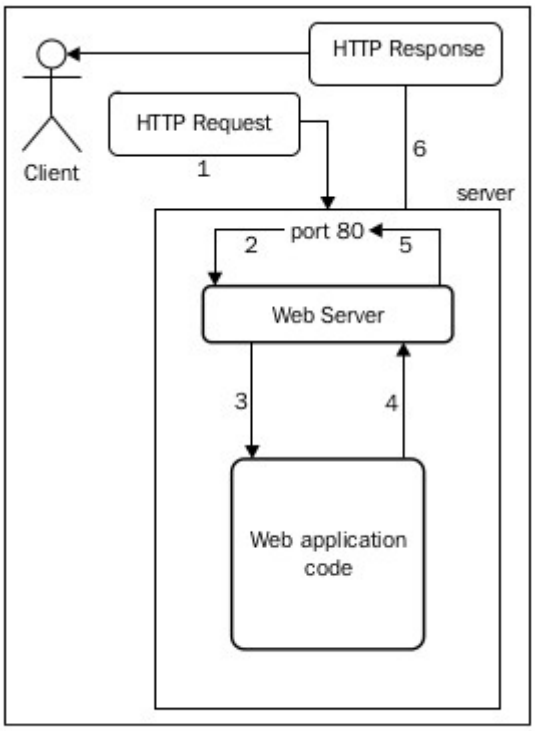
1. The client (browser) sends an HTTP request (GET or POST) to the server.

2. The server listens on a specific port (e.g., 80/443) and captures the request.

3. The web server determines which web application or resource should handle the request, based on the URL path or domain.

4. The web application processes the request (business logic, database queries, etc.) and generates a response.

5. The web server sends the generated response (usually HTML, JSON, or files) back to the client.



Request-response flow on the server side

6. The browser displays the response to the user, completing the HTTP request-response cycle.

b) How a web server remembers a web client between requests?

Solution:

A web server remembers a client between requests by maintaining an HTTP session. Since HTTP is a stateless protocol, the server needs a way to identify which requests belong to which client.

The most common method is through cookies:

- When the server first responds, it includes a Set-Cookie header.
- The client's browser stores this cookie, which contains a small piece of data unique to that session.
- For every subsequent request, the browser automatically sends the cookie back in the Cookie header.

This back-and-forth exchange allows the server to recognize the client and maintain a stateful conversation until the session ends.

c) Explain different ways of including CSS to a HTML page.

Solution:

Three ways of including CSS

1. ***Inline CSS***

   CSS is applied directly within an HTML element using the `style` attribute. It is used for quick, single-element styling. Example:

   ```
   <p style="color: red; font-size: 16px;">Hello</p>
   ```

2. ***Internal CSS***

   CSS is written inside a `<style>` tag within the `<head>` section of the HTML page. It is used for styling a single HTML page. Example:

   ```
   <style>
       p { color: blue; font-size: 16px; }
   </style>
   ```

3. ***External CSS***

   CSS is placed in a separate `.css` file and linked using the `<link>` tag. It is used for styling multiple pages consistently. Example:

   ```
   <link rel="stylesheet" href="styles.css">
   ```

4. a) Mention some benefits of OOP in the context of web application development.

Solution:

Object-oriented programming (OOP) provides several benefits in web application development.

1. Encapsulation allows combining data and behavior into classes, protecting sensitive information and controlling access through methods.

2. Code re-usability is improved, as classes and traits can be reused across different parts of the application, reducing duplication.

3. Inheritance enables new classes to extend existing ones, promoting consistency and saving development time.

4. Polymorphism allows functions to work with objects of different classes that implement the same interface, increasing flexibility.

Overall, OOP enhances maintainability, making it easier to organize, debug, and extend web applications.

b) Write a very simple class in PHP programming language, with one private variable and its setter and getter functions. Also, give an example of using the class.

Solution:

```
<?php
class Person {
    private $name;
    public function setName($name) {
        $this->name = $name;
    }
    public function getName() {
        return $this->name;
    }
}

$person = new Person();
$person->setName("Alice");
echo $person->getName();
?>
```

Output

```
Alice
```

c) What does it mean to autoload classes in PHP? Explain the concepts of inheritance, interfaces and traits in the context of OOP in PHP.

Solution:

AutoLoading

Autoloading in PHP is a feature that automatically loads class files when a class is first used, without needing manual `require` or `include`.

Inheritance

Inheritance allows a class to reuse the properties and methods of another class. The class that inherits is called a ***child class***, and the class being inherited from is called a ***parent class***. This helps reduce code duplication and create a hierarchical relationship between classes.

For example, a `Customer` class could inherit from a `Person` class, automatically gaining its properties like `name` and `email`.

Interfaces

Interfaces define a contract that a class must follow. They declare ***methods without implementation***, and any class implementing an interface must provide concrete implementations for all its methods. This ensures that different classes can be used interchangeably as long as they follow the same interface.

For instance, both `BasicCustomer` and `PremiumCustomer` could implement a `Payer` interface with a `pay()` method, allowing a function to work with any payer without caring about the specific class.

Traits

Traits allow you to reuse code across multiple classes without using inheritance. Unlike interfaces, traits ***can include method implementations*** and properties. They are especially useful when multiple classes need the same functionality, but do not share a parent class.

For example, a `Unique` trait could provide ID management for both `Customer` and `Book` classes, allowing them to share code without creating a complex inheritance structure.\

Section B

5. a) Discuss the various reasons you need to test for developing Web App.

Solution:

Reasons for Testing in Web Application Development

1. ***Correctness*** – To ensure the web app's features work as per requirements.
2. ***Performance*** – To verify speed, response time, and handling of multiple users.
3. ***Compatibility*** – To confirm the app works across browsers, devices, and OS.
4. ***Usability & Accessibility*** – To check user-friendliness and support for all users, including those with disabilities.
5. ***Security*** – To protect against unauthorized access, hacking, and data leaks.

b) Explain the AEIOU principles of application testing.

Solution:

The ***AEIOU principles*** provide a simple mnemonic to guide comprehensive application testing. Each letter represents a key aspect to focus on:

1. ***A – Accessibility:*** Ensure the application is ***accessible to all users***, including people with disabilities. Test for screen readers, keyboard navigation, color contrast, and compliance with accessibility standards (like WCAG).
2. ***E – Efficiency:*** Check if the application ***performs tasks quickly and uses resources optimally***. Includes testing for load times, CPU/memory usage, and responsiveness under different conditions.
3. ***I – Integrity:*** Verify the ***accuracy and reliability of data***. Ensure there is no data corruption, unauthorized data changes, or loss during transactions.
4. ***O – Operability:*** Assess how ***easy and convenient it is to operate the application***. Involves usability testing, user interface intuitiveness, and error handling.
5. ***U – Understandability:*** Evaluate whether the application is **clear and easy to understand** for users. Includes meaningful error messages, documentation, help guides, and logical workflows.

c) What types of tests are performed during the lifecycle of a Web App. Explain.

Solution:

Types of Tests During a Web App Lifecycle

1. ***Non-functional Testing***

    1. ***Performance Testing:*** Measures speed, availability, and stability under load.
    2. ***Load Testing:*** Check how the system behaves under a specific, predefined load.
    3. ***Stress Testing:*** Push the system beyond its limits to see if it crashes or recovers.
    4. ***Usability & Compatibility Testing:*** Ensures user-friendliness and works across browsers/devices.
    5. ***Accessibility Testing:*** Ensure people with disabilities or limited configurations can use the app.
    6. **Security Testing:** Detects vulnerabilities like XSS, CSRF, or SQL injection.

2. **Functional Testing**

    1. ***Unit Testing:*** Checks individual modules for correct functionality.
    2. ***Integration Testing:*** Ensures different modules work together properly.
    3. ***System Testing:*** End-to-end testing of the entire web application after unit and integration testing.
    4. ***Acceptance Testing:*** Validates the app meets client/business requirements before release.

6. a) Compare among MySQL, MySQLi and PDO extensions of PHP for interacting with databases.

Solution:

Feature	***MySQL***	***MySQLi***	***PDO***
Database Support	Only MySQL	Only MySQL	Multiple databases (MySQL, PostgreSQL, SQLite, etc.)
API Type	Procedural	Procedural & Object-Oriented	Object-Oriented only
Prepared Statements	Not supported	Supported	Supported (safer against SQL injection)
Security	Basic	Improved security with prepared statements	High security with prepared statements and parameter binding
Flexibility	Limited	Better, MySQL-specific features	Very flexible for cross-database applications
Error Handling	Manual	Supports error reporting modes	Exception handling with try… catch

b) When is the right time to start designing the database in web application development process? Why do you think so?

Solution:

The right time to start designing the database is during the ***planning and requirement analysis phase***, before writing any application code. The reasons behind it are –

1. ***Data Structure Definition:*** The database design determines how data is organized, related, and stored. Without it, application logic may become inconsistent.
2. ***Prevents Redundancy & Errors:*** Early design avoids data duplication, inconsistent relationships, and integrity issues.
3. ***Supports Application Features:*** Understanding the data helps developers plan queries, relationships, and functionality effectively.

4. ***Improves Performance:*** Proper indexing, normalization, and relationships can be planned ahead, improving scalability and speed.
5. ***Facilitates Collaboration:*** Clear database schema helps backend developers, frontend developers, and designers work consistently.

c) What is the MVC design pattern? Why some of the most popular web application development frameworks use MVC?

Solution:

MVC (Model-View-Controller)

The MVC design pattern is an architectural pattern that separates an application into three interconnected components:

1. ***Model*** – Manages the data and business logic of the application. It interacts with the database and performs operations like ***fetching, updating,*** or ***deleting data.***
2. ***View*** – Handles the ***presentation layer***. It displays data to the user, usually as HTML pages, and formats the output without containing business logic.
3. ***Controller*** – Acts as an ***intermediary*** between the model and view. It ***processes incoming requests***, interacts with models to get data, and passes it to views for display.

Why popular frameworks use MVC:

1. ***Separation of concerns:*** Keeps data, business logic, and presentation separate, making the code easier to maintain.
2. ***Re-usability:*** Models and views can be reused in different parts of the application.
3. ***Testability:*** Easier to unit test individual components without affecting others.
4. ***Scalability:*** Facilitates managing larger projects with multiple developers.
5. ***Clear workflow:*** Simplifies development by clearly defining responsibilities for handling requests, processing data, and rendering output.

7. a) For communicating with various RDBMS, what APIs are available in PHP? Discuss advantages and disadvantages of them.

Solution:

PHP provides following APIs to communicate with Relational Database Management Systems:

1. ***MySQL Extension***

   Original PHP extension for MySQL databases (deprecated as of PHP 5.5 and removed in PHP 7).

   Advantages:

   1. Simple and easy to use for basic operations.
   2. Lightweight for small projects.

   Disadvantages:

   1. Deprecated and no longer supported.
   2. Does not support prepared statements (security risk for SQL injection).
   3. Only works with MySQL, not portable to other databases.

2. ***MySQLi (MySQL Improved)***

   Modern replacement for MySQL extension, supports MySQL 4.1+ features.

   Advantages:

   1. Supports prepared statements (prevents SQL injection).
   2. Offers both ***procedural*** and ***object-oriented*** APIs.
   3. Supports transactions, multi-query execution, and enhanced debugging.

   Disadvantages:

   1. Limited to MySQL databases only.
   2. More complex than old mysql_* functions.

3. ***PDO (PHP Data Objects)***

   PDO is a PHP extension that provides a ***uniform interface*** for accessing different types of databases (MySQL, PostgreSQL, SQLite, etc.) and allows developers to ***connect, query,***

and manipulate databases securely and efficiently, with support for ***prepared statements***, transactions, and error handling.

Advantages	***Disadvantages***
1. Database-independent, making code portable. 2. Supports prepared statements (secure against SQL injection). 3. Object-oriented interface with flexible error handling.	1. Slightly slower than native extensions for very large queries. 2. Does not support some database-specific features natively.

b) How query() and exec() methods differ in PDO?

Solution:

query() vs exec()

Feature	***`query()`***	***`exec()`***
Type of query	SELECT	INSERT, UPDATE, DELETE
Return value	PDOStatement object	Number of affected rows
Fetching data	Yes, using fetch()/fetchAll()	No
Error handling	Throws exception on failure	Throws exception on failure
Use case	Used for retrieving data	Used for modifying data

c) What is composer? Explain the basic structure of a composer.json file.

Solution:

Composer

Composer is a dependency management tool for PHP that allows developers to declare the libraries their project depends on and automatically installs and updates them.

Composer ensures that the correct versions of libraries are installed and helps manage project dependencies efficiently.

Basic structure of a `composer.json` file:

The key sections are:

1. **`name`** – The name of your project, usually in `vendor/project` format.
2. **`description`** – A short description of the project.
3. **`require`** – Lists the PHP packages or libraries the project depends on, along with version constraints.
4. **`autoload`** – Defines how your project classes should be loaded automatically, e.g., PSR-4 autoloading.
5. **`require-dev`** – Optional. Lists packages needed only for development, like testing frameworks.
6. **`scripts`** – Optional. Defines scripts that can be run before or after certain Composer events.
7. **`config`** – Optional. Configures Composer behavior, like preferred installation methods.

8. a) Discuss how the following attacks are performed on an insecure Web App and how can we protect against those attacks:

(i) SQL Injection

Solution:

Performed: Attacker injects malicious SQL code into input fields (e.g., login form, search box) that the app concatenates directly into queries. This can bypass authentication, read/modify/delete data.

Protection: Use prepared statements/parameterized queries, validate input, enforce least-privilege DB accounts, and hide error messages.

(ii) File upload attack

Solution:

Performed: Attacker uploads a malicious file (e.g., PHP/ASP web shell) to the server, then executes it via direct URL to gain control.

Protection: Store uploads outside web root, validate MIME type and content, rename files, restrict executable permissions, use antivirus scanning.

(iii) Cross-site request forgery

Solution:

Cross-Site Request Forgery (CSRF)

- **Attack:** Attacker tricks a logged-in user into sending unintended requests like money transfer using their cookies/session.
- **Example:** Malicious link like `bank.com/transfer?to=attacker&amt=1000` auto-executes if user is logged in.
- **Protection:** Use POST not GET for state changes, anti-CSRF tokens, and SameSite cookies.

b) What measure should you take to secure your website?

Solution:

Measures to Secure a Website

1. ***Input validation & sanitization*** – Validate all user inputs and escape special characters to prevent SQL Injection and XSS.
2. ***Use HTTPS & secure cookies*** – Encrypt communication, set cookies with `HttpOnly`, `Secure`, and `SameSite` flags.
3. ***Authentication & access control*** – Enforce strong passwords, multi-factor authentication, and least-privilege roles.
4. ***Session security*** – Implement anti-CSRF tokens, session timeouts, and re-authentication for sensitive actions.
5. ***Regular patching & monitoring*** – Keep frameworks and servers updated, log security events, and scan for vulnerabilities.

Year - 2021

Section-A

1.(a) What are the different types of web-based systems? Provide examples for each.

Solution:

Document-Centric	***Interactive Web Applications***
Description: Predecessors of modern web apps; static HTML documents stored on a web server and delivered to clients. **Features:** Updated manually, simple, stable, short response time, but can lead to outdated information or inconsistencies. **Examples:** Static homepages, small business websites.	**Description:** Use CGI and HTML forms to generate dynamic pages. **Features:** User interaction, dynamically generated content. **Examples:** Virtual exhibitions, news sites, timetable systems.
Transactional	***Workflow-Based***
Description: Highly interactive, data-driven; allow users to update information efficiently. **Features:** Consistent handling of large content, often tied to databases. **Examples:** Online banking, e-commerce, booking systems.	**Description:** Automate workflows within or between organizations. **Features:** Require structured processes, robust and flexible, handle complex interactions. **Examples:** B2B e-commerce solutions, e-government portals, healthcare workflow systems.
Collaborative	***Social***
Description: Support group work and communication. **Features:** Shared information, workspaces, high degree of communication.	**Description:** Connect users with similar interests and facilitate community interaction. **Features:** Identity sharing, interest-based connections.

Examples: Wikis, Google Meet, Google Classroom, scheduling systems.	**Examples:** Blogs, Facebook, Friendster.
Portal-Oriented	***Ubiquitous***
Description: Central hubs providing access to multiple, heterogeneous sources. **Features:** Single point of access, specialized information delivery. **Examples:** Business portals, healthcare portals, marketplace portals.	**Description:** Accessible everywhere, seamlessly integrated into users' lives across devices. **Features:** Cross-platform compatibility Responsive design Offline functionality Cloud-based infrastructure Location awareness and personalization **Examples:** Mobile banking apps, cloud-based productivity apps.
Semantic	
Description: Present data in a machine-readable form to support knowledge management. **Features:** Facilitates content syndication, recommender systems, and more intelligent search. **Examples:** Semantic search engines, intelligent recommendation platforms.	

(b) What agility principles are adopted by a good Web Engineering Team?

Solution:

There are ***Twelve Agility Principles*** which are based on Agility Manifesto are adopted by a good Web Engineering Team:

1. ***Customer Satisfaction*** – Deliver valuable software that meets customer needs.
2. ***Embrace Changing Requirements*** – Adapt to new or evolving requirements even late in development.
3. ***Frequent Delivery*** – Deliver working increments of the web application regularly.
4. ***Promote Collaboration*** – Foster close collaboration between developers and stakeholders.

5. ***Motivated Individuals*** – Build a motivated team, provide support, and trust them to get the job done.
6. ***Maintain a Constant Pace*** – Ensure sustainable development without burnout.
7. ***Face-to-Face Communication*** – Prefer direct communication over lengthy documentation.
8. ***Measure Progress*** – Track progress based on working software, not just tasks completed.
9. ***Technical Excellence*** – Continuously improve technical skills and maintain high code quality.
10. ***Simplicity*** – Focus on simplicity, avoiding unnecessary work.
11. ***Self-Organized Team*** – Allow teams to organize themselves for better results.
12. ***Continuous Improvement*** – Regularly reflect on processes and practices to improve efficiency and quality.

2.(a) Emphasize the importance of communication activity in WebE process. What technique is used for communication with the stakeholders?

Solution:

Importance of Communication Activity in Web Engineering (WebE) Process

1. Communication activity provides the WebE team with an organized way of ***formulating***, ***eliciting requirements*** from stakeholders and ***negotiate*** with them.
2. It acts as the ***entry point*** for the process flow.
3. It ensures that both stakeholders and developers have a ***shared understanding*** of the WebApp's purpose, scope, and constraints.

Key Tasks:

1. ***Formulation –*** Ask and answer fundamental questions about the WebApp and its business context.
2. ***Elicit requirements*** – that will serve as the foundation for all subsequent activities.
3. ***Negotiate –*** needs against the realities of time, resources, and technology.

Techniques Used for Communication with Stakeholders

1. Traditional Focus Group

2. Electronic Focus Group
3. Iterative Surveys
4. Exploratory Survey
5. Scenario Building

(b) Discuss 'use case diagram' as web app analysis tools.

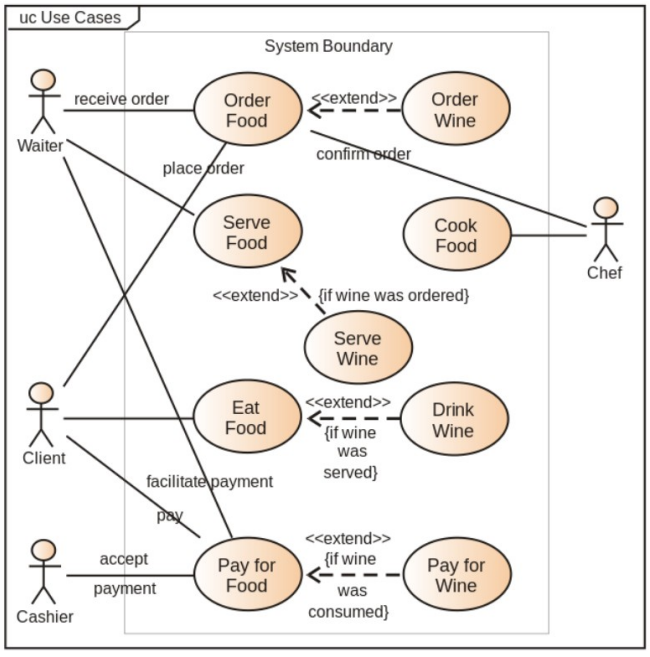
Solution:

Use Case Diagram

A use case diagram is a behavioral diagram in UML that represents the ***functional requirements*** and shows ***interactions between users (actors) and the system***, helping developers and stakeholders understand what the system should do.

Importance in Web App Analysis:

1. ***Requirement Gathering:*** Captures functional requirements in an easy-to-understand way.
2. ***Stakeholder Communication:*** Provides a simple visual tool for discussion with non-technical stakeholders.
3. ***System Boundary Definition:*** Shows what lies inside the system (use cases) and outside (actors).
4. ***Identifies Actors & Roles:*** Clarifies who will interact with the system (e.g., client, cashier, waiter, chef).
5. ***Clarifies Functionality:*** Highlights core system functions and optional/extended behaviors.
6. ***Supports Incremental Development:*** Helps plan increments of functionality (e.g., food order first, wine order later).



Example (Restaurant Use Case Diagram)

The diagram you provided illustrates:

- ***Actors:*** Waiter, Client, Chef, Cashier.

- ***Use Cases:*** Order Food, Cook Food, Serve Food, Eat Food, Pay for Food, etc.
- ***Extensions:*** Optional behaviors like *Order Wine*, *Serve Wine*, *Drink Wine*, *Pay for Wine*.
- ***Relationships:*** Communication between actors and system, with `<<extend>>` relationships to model optional behavior.

3.(a) Define composer.

Solution:

Composer

Composer is a dependency management tool for PHP that allows developers to declare the libraries their project depends on and automatically installs and updates them.

Composer ensures that the correct versions of libraries are installed and helps manage project dependencies efficiently.

(b) Explain important directories used in a common Laravel application.

Solution:

Important directories in a common Laravel application

1. **app/** – Contains core application code: models, controllers, and services.
2. **bootstrap/** – Bootstraps the application and loads configuration.
3. **config/** – Holds configuration files for database, mail, cache, etc.
4. **database/** – Contains migrations, seeders, and factories for database management.
5. **public/** – Web root; contains `index.php` and public assets (CSS, JS, images).
6. **resources/** – Stores views (Blade templates), language files, and raw assets.
7. **routes/** – Defines all application routes (web.php, api.php).
8. **storage/** – Stores logs, cached files, and uploads.
9. **vendor/** – Contains Composer-managed dependencies.

(c) What are named Routes in Larevel?

Solution:

Named Routes in Laravel

Named routes allow you to assign a **name** to a route, so you can refer to it easily elsewhere in your application instead of using the URL directly. This improves maintainability because if the URL changes, you only need to update it in one place.

Examples:

```
Route::get('/books', [BookController::class,'index'])→name('books.index');
```

Usage in views or controllers:

```
<a href="{{ route('books.index') }}">Books</a>
```

4. (a) How does spl_autoload_register allow for more flexibility and customization in autoloading classes than_autoload?

Solution:

__autoload()

- Only ***one function*** can be defined.
- PHP calls it automatically when an unknown class is used.
- Limited flexibility — you cannot have multiple autoload strategies.

spl_autoload_register()

- Allows you to ***register multiple autoload functions***.
- PHP will try them ***in the order they were registered*** until the class is found.
- Supports ***named functions, static methods, or anonymous functions***.
- Makes it easy to customize loading rules (different directories, namespaces, libraries).

(b) What is trait in PHP, and how is it different from a class or interface?

Solution:

Trait

In PHP, a ***trait*** is a mechanism for reusing code across multiple classes. Traits can contain methods and properties, but unlike classes, they ***cannot be instantiated*** on their own. They are “injected” into classes using the `use` keyword, allowing different classes to share the same functionality without requiring inheritance.

The difference from ***classes*** is that a class can be instantiated and represents an object blueprint, whereas a trait is just a reusable piece of code.

The difference from ***interfaces*** is that interfaces only define method signatures that a class must implement, but traits provide actual method implementations.

(c) What is the difference between require() and require_once() function?

Solution:

require()

This will do the same as include, but PHP will throw an error instead of a warning if the file is not found.

require_once()

This works the same as require, but it will include the file only the first time that it is invoked. Subsequent calls will be ignored.

Section-B

5.(a) Why testing is necessary for web application?

Solution:

Reasons for Testing in Web Application Development

1. ***Correctness*** – To ensure the web app's features work as per requirements.
2. ***Performance*** – To verify speed, response time, and handling of multiple users.
3. ***Compatibility*** – To confirm the app works across browsers, devices, and OS.
4. ***Usability & Accessibility*** – To check user-friendliness and support for all users, including those with disabilities.
5. ***Security*** – To protect against unauthorized access, hacking, and data leaks.

(b) How does Integration test differ from Acceptance test?

Solution:

Aspect	***Integration Test***	***Acceptance Test***
Purpose	To check if different modules of the app work together correctly.	To verify the app meets business requirements and is ready for release.
Performed By	Developers or QA team.	End-users or clients.
Focus	Interaction between modules and data flow.	Overall functionality, usability, and requirements fulfillment.
Timing	Done during development after unit testing.	Done at the end of development before deployment.

(c) Briefly explain the idea of 'code coverage'.

Solution:

Code Coverage

Code coverage is a measure of how much of the source code is executed when the application is tested.

It helps to identify untested parts of the code so developers can write additional tests.

Types:

- ***Function Coverage:*** Tests all functions/methods.
- ***Statement Coverage:*** Tests all statements/lines of code.
- ***Branch Coverage:*** Tests all decision points (if/else, loops).

Benefit of code coverage is that it ensures more thorough testing and reduces the chance of hidden bugs.

6.(a) What are the three components of the MVC architecture, and what is the role of each component?

Solution:

MVC (Model-View-Controller)

The MVC design pattern is an architectural pattern that separates an application into three interconnected components:

4. ***Model*** – Manages the data and business logic of the application. It interacts with the database and performs operations like ***fetching, updating,*** or ***deleting data.***
5. ***View*** – Handles the ***presentation layer***. It displays data to the user, usually as HTML pages, and formats the output without containing business logic.
6. ***Controller*** – Acts as an ***intermediary*** between the model and view. It ***processes incoming requests***, interacts with models to get data, and passes it to views for display.

(b) How does a router Interact with a controller in a MVC web application?

Solution:

In an MVC web application, the router and controller interact as follows:

1. **Request Handling:**

   The ***router*** receives the incoming HTTP request (URL, method, parameters). It determines which ***controller*** and ***method*** should handle the request based on the route definitions.

2. **Parameter Passing:**

The router extracts any URL parameters (e.g., `/books/5`) and passes them to the corresponding controller method.

3. **Controller Invocation:**

   The router instantiates the specified ***controller*** (if not already instantiated). It calls the appropriate method on the controller, providing any request data or parameters.

4. **Response Return:**

   The controller processes the request (interacting with models if needed) and returns a ***response*** (HTML, JSON, etc.). The router then sends this response back to the client.

(c) What is Object-Relational Mapping (ORM)? How It is done in Laravel MVC Framework?

Solution:

Object-Relational Mapping (ORM)

ORM is a technique that allows developers to interact with a ***database using objects*** instead of writing raw SQL queries. It ***maps database tables*** to ***classes*** and ***rows*** to ***objects***, making data manipulation easier and more intuitive in object-oriented programming.

ORM in Laravel MVC Framework:

1. ***Models represent database tables*** – each table corresponds to a model class.
2. ***Data operations are done through models*** – you can fetch, insert, update, or delete records using object-oriented methods.
3. ***Relationships are defined in models*** – such as one-to-many, many-to-many, allowing easy navigation between related tables

7. (a) What are the advantages of using prepared statements in PDO? Rewrite the code below so that it uses PDO prepare statement.

```
$db = new PDO();
$author = 'Adam Smith';
$row = $db->query("SELECT * FROM book where author "'.$author.'";");
```

Solution:

Advantages of using prepared statements in PDO

1. ***Prevents SQL Injection:*** User inputs are safely handled, avoiding malicious code execution.
2. ***Re-usability:*** The same query can be executed multiple times with different parameters.
3. ***Improved Performance:*** The database can optimize the query execution plan since the query structure is fixed.
4. ***Cleaner Code:*** Separation of SQL and data makes the code easier to read and maintain.
5. ***Automatic Data Handling:*** PDO automatically handles quoting and escaping of values.

Rewriting the given code using PDO prepared statements:

```
$db = new PDO('mysql:host=127.0.0.1;dbname=bookstore', 'root', '');
$author = 'Adam Smith';

$query = 'SELECT * FROM book WHERE author = :author';
$statement = $db→prepare($query);

$statement->bindValue(':author', $author);
$statement->execute();

$row = $statement->fetchAll(PDO::FETCH_ASSOC);
var_dump($row);
```

b) What is the possible use case for anonymous function?

Solution:

Anonymous Function

An ***anonymous function***, also called a ***lambda function***, is a function in PHP that has no name and can be stored in a variable or passed as an argument to another function.

It is useful for creating small, self-contained pieces of logic that are only needed temporarily, without cluttering the global namespace.

- A common use case is ***as a callback*** in array operations, such as `array_map`, or `array_filter`, where the function is applied to each element of an arr

Example:

```
$numbers = [1, 2, 3, 4, 5];
$squared = array_map(function($n) { return $n * $n; }, $numbers);
print_r($squared);
```

In this example, the anonymous function squares each number in the array without needing a separate named function.

(c) Explain how composer is used for dependency management in PHP?

Solution:

Composer and Dependency Management in PHP

Composer is a dependency manager for PHP that allows you to declare, install, and manage libraries (packages) your project depends on.

How it works

- You create a `composer.json` file in your project specifying the packages you need.
- Composer downloads the packages and their dependencies into a `vendor/` directory.
- It also generates an `autoload.php` file to automatically load classes from those packages.

8.(a) list out common artisan commands used in Laravel.

Solution:

- `php artisan serve` → start the server
- `php artisan make:controller ControllerName` → create controller
- `php artisan make:model ModelName` → create model
- `php artisan make:migration migration_name` → create migration
- `php artisan migrate` → run migration
- `php artisan route:list` → check routes

(b) State the difference between get and post method.

Solution:

GET

This asks the receiver about something, and the receiver usually sends this information back. The most common example is asking for a web page, where the receiver will respond with the HTML code of the requested page.

POST

This means that the sender wants to perform an action that will update the data that the receiver is holding. For example, the sender can ask the receiver to update his profile name.

(c) Why are migrations important?

Solution:

Migrations are important because they ***manage and version-control the database schema*** in a consistent way. Instead of manually creating tables or altering them, migrations allow developers to:

1. ***Create, modify, and delete tables*** through code.
2. ***Share database structure*** among team members easily.
3. ***Roll back changes*** if something goes wrong.
4. Keep ***database in sync*** across different environments (local, staging, production).

Year - 2022

Section-A

1. a) What are the different types of web-based systems? Provide examples for each.

Solution:

Document-Centric	***Interactive Web Applications***
Description: Predecessors of modern web apps; static HTML documents stored on a web server and delivered to clients. **Features:** Updated manually, simple, stable, short response time, but can lead to outdated information or inconsistencies. **Examples:** Static homepages, small business websites.	**Description:** Use CGI and HTML forms to generate dynamic pages. **Features:** User interaction, dynamically generated content. **Examples:** Virtual exhibitions, news sites, timetable systems.
Transactional	***Workflow-Based***
Description: Highly interactive, data-driven; allow users to update information efficiently. **Features:** Consistent handling of large content, often tied to databases. **Examples:** Online banking, e-commerce, booking systems.	**Description:** Automate workflows within or between organizations. **Features:** Require structured processes, robust and flexible, handle complex interactions. **Examples:** B2B e-commerce solutions, e-government portals, healthcare workflow systems.
Collaborative	***Social***
Description: Support group work and communication. **Features:** Shared information, workspaces, high degree of communication.	**Description:** Connect users with similar interests and facilitate community interaction. **Features:** Identity sharing, interest-based connections.

Examples: Wikis, Google Meet, Google Classroom, scheduling systems.	**Examples:** Blogs, Facebook, Friendster.
Portal-Oriented	***Ubiquitous***
Description: Central hubs providing access to multiple, heterogeneous sources. **Features:** Single point of access, specialized information delivery. **Examples:** Business portals, healthcare portals, marketplace portals.	**Description:** Accessible everywhere, seamlessly integrated into users' lives across devices. **Features:** Cross-platform compatibility Responsive design Offline functionality Cloud-based infrastructure Location awareness and personalization **Examples:** Mobile banking apps, cloud-based productivity apps.
Semantic	
Description: Present data in a machine-readable form to support knowledge management. **Features:** Facilitates content syndication, recommender systems, and more intelligent search. **Examples:** Semantic search engines, intelligent recommendation platforms.	

b) What are the unique characteristics that separates web applications from other software applications?

Solution:

Web Apps vs Traditional Software

Attribute	Web Apps	Traditional Software
Network Intensiveness	Relies on internet or network connectivity to function.	Often works offline; network is optional.
Concurrency	Supports multiple users accessing simultaneously.	Usually single-user or limited concurrency.

Unpredictable Load	Traffic can vary drastically; must handle spikes.	Load is generally predictable, installed per system.
Performance	Needs optimization for varying network speeds and server load.	Performance mostly depends on local machine resources.
Availability	Must be available 24/7 for users.	Usually only needs to be available when installed/used.
Data Driven	Frequently interacts with databases and real-time data.	May store data locally; interaction with databases is optional.
Content Sensitive	Content can be dynamic and personalized per user.	Content is mostly static and uniform.
Continuous Evolution	Updated regularly without user intervention.	Updates require user action (installation/upgrade).
Immediacy	Accessible instantly via browser; no installation required.	Requires installation before use.
Security	Faces web-specific threats (XSS, CSRF, SQL injection).	Security is mostly local or system-based.
Aesthetics	UI/UX is critical for user engagement and satisfaction.	UI may be less flexible, often depends on OS standards.

c) What do you understand by semantic web?

Solution:

Semantic Web

The Semantic Web is an extension of the current web that enables data to be understood and processed by machines, not just humans.

It allows information on the web to be structured, linked, and interpreted in a way that computers can reason about it, making web content more intelligent and useful.

Key Points:

- Presents data in a **machine-readable format** using standards like **RDF** and **OWL.**

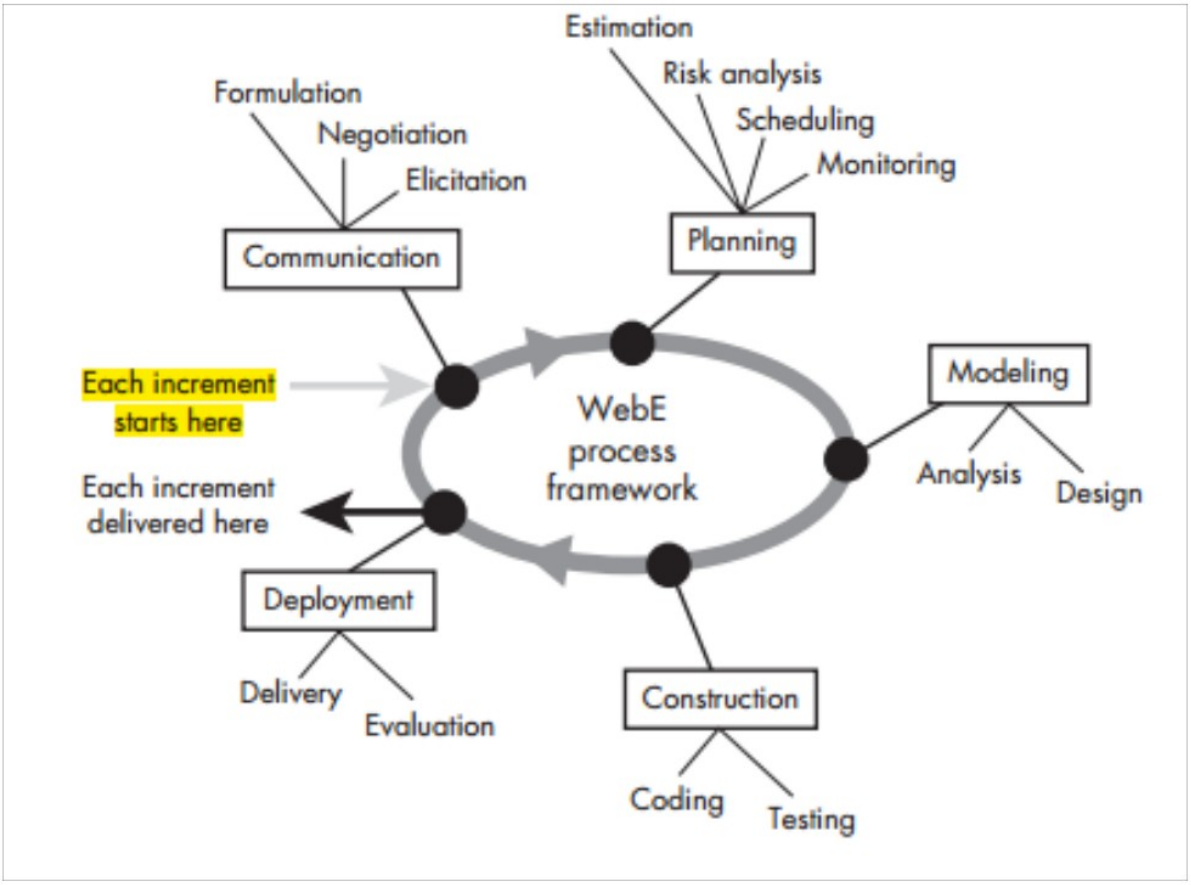
- Facilitates knowledge management, content syndication, enhanced search and recommendation systems.
- Enables applications to ***find relevant information automatically*** and integrate knowledge from multiple sources.

2. a) What is a web engineering process framework? Discuss different components of this process framework.

Solution:

Web Engineering Process Framework

A Web Engineering (WebE) process framework is a structured model that defines the set of ***activities, tasks, and workflows*** involved in developing, deploying, and maintaining high-quality web applications.



Components of the WebE Process Framework

1. **Communication**

- Entry point of the process.
- Involves ***formulation, negotiation, and elicitation*** of requirements.
- Ensures stakeholder needs are clearly understood.

2. **Planning**
    - Defines the roadmap for development.
    - Activities include ***estimation, risk analysis, scheduling, and monitoring***.
    - Ensures resources and timelines are well managed.
3. **Modeling**
    - Focuses on ***analysis and design***.
    - Creates models (data, navigation, architecture, interface) to better understand requirements and solutions.
4. **Construction**
    - Actual ***coding and testing*** of the web application.
    - Ensures functionality works as expected and errors are minimized.
5. **Deployment**
    - ***Delivery*** of the web application increment to the customer.
    - Followed by ***evaluation and feedback*** to refine the system.

b) What is the importance of communication activity' in defining the overall web application requirements?

Solution:

Importance of Communication Activity in Web Engineering (WebE) Process

4. Communication activity provides the WebE team with an organized way of ***formulating***, ***eliciting requirements*** from stakeholders and ***negotiate*** with them.
5. It acts as the ***entry point*** for the process flow.
6. It ensures that both stakeholders and developers have a ***shared understanding*** of the WebApp’s purpose, scope, and constraints.

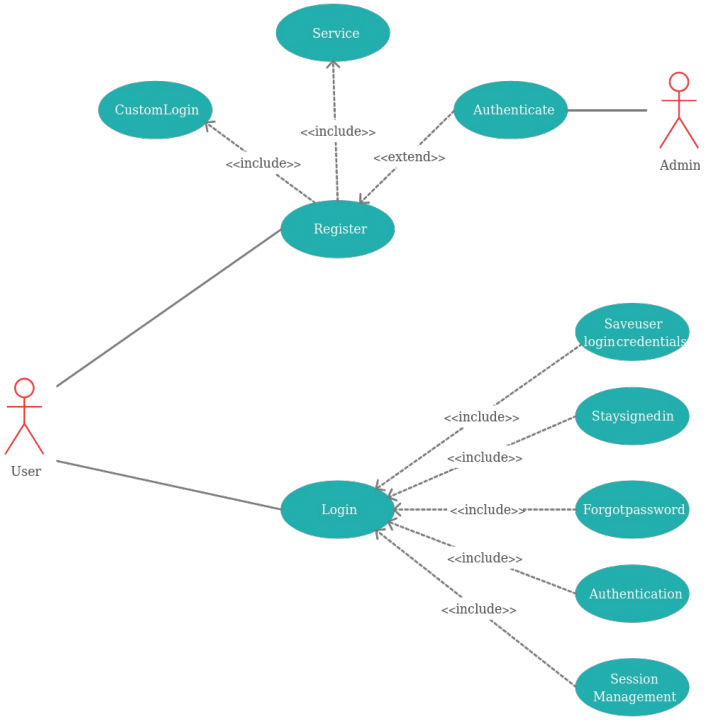
Key Tasks:

1. ***Formulation –*** Ask and answer fundamental questions about the WebApp and its business context.
2. ***Elicit requirements*** – that will serve as the foundation for all subsequent activities.
3. ***Negotiate –*** needs against the realities of time, resources, and technology.

c) Construct a use case scenario for 'User Login' functionality for a web application.

Solution:

Use Case: User Login



- ***Actors:*** User, Admin
- ***Preconditions:*** User must be registered with valid credentials.
- ***Main Flow:*** User enters username/email & password → System authenticates → On success, user is redirected to dashboard.
- ***Alternate Flows:*** Invalid credentials (error message), Forgot password (reset link), Stay signed in (auto-login).
- ***Post-condition:*** User is logged in and can access the application's features.

3. a) Discuss various categories of risks that can delay/hamper the development of a web app.

Solution:

Categories of Risks

1. ***People Risks*** – Risks arising from human factors such as lack of skills, miscommunication, or poor coordination.
   Example: *Developer unfamiliar with new technology causing delays.*

2. ***Product Risks*** – Risks linked to the web app itself, such as content errors, functional failures, security loopholes, or performance issues.
   Example: *App crashes under high load.*
3. ***Process Risks*** – Risks due to flaws in the chosen development process or methodology.
   Example: *Incomplete requirements elicitation leading to rework.*

b) How can you evaluate those risks and prepare for them?

Solution:

Risk Evaluation & Preparation

Risks in web app development can be evaluated by estimating:

- ***Probability*** (percentage) of the risk occurring.
- ***Impact*** (3 = High/ 2 = Medium/ 1 = Low) if the risk occurs.

Based on this, a risk evaluation table can be created, followed by a contingency plan for each risk.

Example Risk Evaluation Table

Risk Type	**Probability (P)**	**Impact (I)**	**Risk Evaluation ($\Re = P \times I$)**	**Contingency Plan**
People Risk (lack of skill)	60%	3	180	Provide training, assign mentors
Product Risk (performance issue)	80%	3	240	Conduct load testing, optimize code
Process Risk (poor requirements)	50%	2	100	Frequent stakeholder reviews, use prototyping

From the table, we can see performance issue has more risk evaluation than other risks. Hence it should be prioritized first.

Preparation (Contingency Planning)

1. ***Identify*** high-probability & high-impact risks.

2. ***Plan*** preventive measures (training, testing, reviews).
3. ***Prepare fallback strategies*** (extra resources, schedule buffers, backup systems).

c) Discuss the technique of developing a schedule for a web application.

Solution:

WebApp Project Scheduling

WebApp project scheduling is the activity of allocating estimated effort for specific WebE tasks across a planned timeline for building each increment.

There are ***two levels*** of scheduling:

1. Macroscopic Schedule

- High-level schedule.
- Identifies all WebApp increments.
- Projects deployment dates for each increment.
- Provides an overall view of the project timeline.

Example: Deploy increment-1 in Week 4, increment-2 in Week 8, etc.

2. Incremental Schedule

- Breaks down each increment into fine-grained tasks.
- Focuses on detailed activities for design, development, and testing.
- Provides step-by-step guidance for the team.

Example of Incremental Tasks (for interface design):

- Design the interface.
- Sketch page layout for space design page.
- Review layout with stakeholders.
- Design space layout navigation mechanisms.
- Design “drawing board” layout.
- Develop procedural details for:

    - graphical wall layout
    - wall length computation & display
    - graphical window layout
    - graphical door layout
- Design selection mechanisms for security system components (sensors, cameras, etc.).
- Develop procedural details for security system layout.
- Conduct walkthroughs with the team/stakeholders.

4. a) Changes are inevitable during different stages of web app development. How would you address those changes in a systematic way?

Solution:

Changes are inevitable during web app development, and they can be managed systematically through a ***Change Management Process***:

1. ***Change Request:*** Any proposed change is formally documented and submitted.
2. ***Impact Analysis:*** Assess how the change affects project timeline, cost, code, and functionality.
3. ***Approval:*** Changes are reviewed and approved by stakeholders or a change control board.
4. ***Implementation & Testing:*** Approved changes are implemented and tested to ensure no existing features break.
5. ***Documentation & Communication:*** Update project documents and inform all team members.

b) Why modeling is necessary?

Solution:

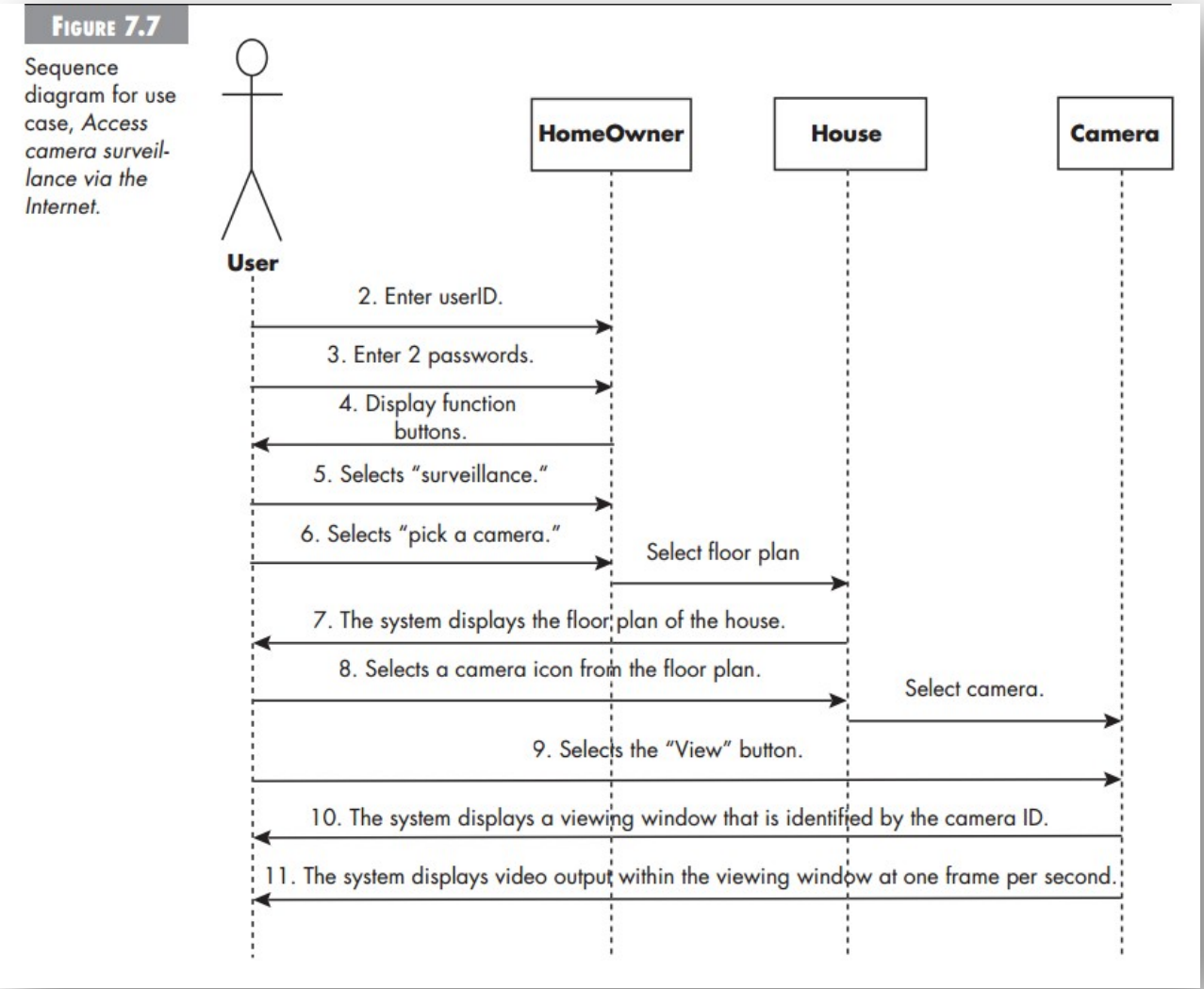
Modeling is necessary in web app development because it helps to:

1. ***Visualize the system:*** Shows how different components interact, making complex systems easier to understand.

2. ***Plan changes effectively:*** Helps anticipate the impact of modifications before implementation.
3. ***Reduce errors:*** Identifies design flaws or conflicts early in development.
4. ***Improve communication:*** Provides a clear representation for developers, testers, and stakeholders.
5. ***Guide development:*** Serves as a blueprint for coding, testing, and deployment.

c) Suppose you are developing a web-based system for A company named "Safe Home" that sales security products and surveillance service. Using their web app user will be able to access cameras installed in their home from internet and check for any possible break-in. Draw a UML sequence diagram for the use-case-Access camera surveillance via the Internet." Property of Seminar Library Dept. of Computer Science a Engineering University of Rajshahi.

Solution:



Section-B

5. a) What are the three components of the MVC architecture, and what is the role of cach component?

Solution:

MVC (Model-View-Controller)

The MVC design pattern is an architectural pattern that separates an application into three interconnected components:

7. ***Model*** – Manages the data and business logic of the application. It interacts with the database and performs operations like ***fetching, updating,*** or ***deleting data.***
8. ***View*** – Handles the ***presentation layer***. It displays data to the user, usually as HTML pages, and formats the output without containing business logic.
9. ***Controller*** – Acts as an ***intermediary*** between the model and view. It ***processes incoming requests***, interacts with models to get data, and passes it to views for display.

b) How does a router interact with a controller in a MVC web application?

Solution:

In an MVC web application, the router and controller interact as follows:

5. **Request Handling:**

   The ***router*** receives the incoming HTTP request (URL, method, parameters). It determines which ***controller*** and ***method*** should handle the request based on the route definitions.

6. **Parameter Passing:**

   The router extracts any URL parameters (e.g., `/books/5`) and passes them to the corresponding controller method.

7. **Controller Invocation:**

   The router instantiates the specified ***controller*** (if not already instantiated). It calls the appropriate method on the controller, providing any request data or parameters.

8. **Response Return:**

   The controller processes the request (interacting with models if needed) and returns a ***response*** (HTML, JSON, etc.). The router then sends this response back to the client.

c) What is Object-Relational Mapping (ORM)? How it is done in Laravel MVC Framework?

Solution:

Object-Relational Mapping (ORM)

ORM is a technique that allows developers to interact with a ***database using objects*** instead of writing raw SQL queries. It ***maps database tables*** to ***classes*** and ***rows*** to ***objects***, making data manipulation easier and more intuitive in object-oriented programming.

ORM in Laravel MVC Framework:

1. ***Models represent database tables*** – each table corresponds to a model class.
2. ***Data operations are done through models*** – you can fetch, insert, update, or delete records using object-oriented methods.
3. ***Relationships are defined in models –*** such as one-to-many, many-to-many, allowing easy navigation between related tables

6. a) Explain how hacker can perform SQL injection attack on a website. How we can protect against this attack?

Solution:

Performed: Attacker injects malicious SQL code into input fields (e.g., login form, search box) that the app concatenates directly into queries. This can bypass authentication, read/modify/delete data.

Protection: Use prepared statements/parameterized queries, validate input, enforce least-privilege DB accounts, and hide error messages.

b) Discuss how insecure handling of file upload can leads to your web application getting hacked. How can you protect that?

Solution:

Performed: Attacker uploads a malicious file (e.g., PHP/ASP web shell) to the server, then executes it via direct URL to gain control.

Protection: Store uploads outside web root, validate MIME type and content, rename files, restrict executable permissions, use antivirus scanning.

7. a) What is PDO? Distinguish between query() and exec() method in PDO.

Solution:

PDO

PDO is a PHP extension that provides a ***uniform interface*** for accessing different types of databases (MySQL, PostgreSQL, SQLite, etc.) and allows developers to ***connect, query, and manipulate databases securely and efficiently***, with support for ***prepared statements***, transactions, and error handling.

query() vs exec()

Feature	***query()***	***exec()***
Type of query	SELECT	INSERT, UPDATE, DELETE
Return value	PDOStatement object	Number of affected rows
Fetching data	Yes, using fetch()/fetchAll()	No
Error handling	Throws exception on failure	Throws exception on failure
Use case	Used for retrieving data	Used for modifying data

b) Suppose you have access to a remote database server with the following information:

DB Server Address	***192.168.31.108***
DB User Name	***cse user***
DB Password	***cse@123***
User's database	***books db***

Design a book title search form and implement server-side logic to search for books in the books table (id, isbn, title, author, price, stock) using a partial author name as input, and display the results on a webpage.

Solution:

HTML Form Design:

- Create a simple HTML form with a text input field for the author name.
- Include a submit button that sends the data to the server (using `POST` or `GET` method).

Server-Side Logic Using PDO:

- Establish a connection to the remote database using PDO:

```
$db = new PDO('mysql:
                host=192.168.31.108;
                dbname=books', cse_user', 'cse@123');
```

- Set the default fetch mode to associative arrays for easier data handling.

Using Prepared Statements:

- Prepare a SQL query with a parameter for the author name to prevent SQL injection:

```
SELECT * FROM books WHERE author LIKE :author
```

- Bind the user input to the parameter using wildcards for partial matching: '%input%'.

Execute Query and Fetch Results:

- Execute the prepared statement.
- Use ***fetchAll()*** to retrieve matching rows.

Display Results:

Loop through the fetched rows and display each book's details (id, isbn, title, author, price, stock) in a readable format (e.g., HTML table).

c) What is a Cookie? How you can access the cookie in PHP?

Solution:

Cookie

A ***cookie*** is a small piece of data sent from a web server and stored in the user's web browser. Cookies are used to remember information about the user, such as login details, preferences, or tracking data.

Accessing a Cookie in PHP:

Cookies sent by the browser can be accessed using the `$_COOKIE` superglobal array:

```
if(isset($_COOKIE["username"])) {
    echo "Username: " . $_COOKIE["username"];
} else {
   echo "Cookie not set.";
}
```

8. a) What is the front-end of a web application, and what technologies are commonly used to develop it?

Solution:

Front-end of a web application

The front-end is the user interface (UI) and user experience (UX) portion of a web app, running inside the browser. It handles:

- Displaying content like text, image, videos or collecting user input via buttons, forms.
- Communicating with the back-end via HTTP/HTTPS, APIs.

Common Technologies for Front-End Development

1. ***Core Web Technologies***
    - ***HTML:*** Defines the structure and content of the web page.
    - ***CSS:*** Controls layout, colors, fonts, and overall style.
    - ***JavaScript:*** Adds interactivity and dynamic behavior.
2. ***Frameworks & Libraries***
    - ***JS Libraries:*** React.js.
    - ***Front-end Frameworks:*** Angular, Svelte.
    - ***CSS Frameworks:*** Bootstrap, Tailwind CSS.
3. ***Modern Tools & Practices***
    - ***Responsive Design:*** Media queries, Flexbox, Grid to support desktops, tablets, and mobiles.
    - ***Build Tools:*** Vite, Babel.
    - ***Package Managers:*** npm, Yarn.

b) What do you understand by REST API?

Solution:

REST API

REST API or Representational State Transfer API is a way for web applications to communicate over HTTP. It is stateless, platform-independent, and can return data in formats like JSON or XML.

It allows a client like a browser or mobile app to send requests to a server to create, read, update, or delete data. REST APIs use standard HTTP methods:

- **GET** – Retrieve data
- **POST** – Create data
- **PUT/PATCH** – Update data
- **DELETE** – Remove data

c) Discuss the role of client-side scripting in building web application.

Solution:

Role of Client-Side Scripting

Client-side scripting mainly using ***JavaScript*** runs in the user's browser and makes web pages ***interactive and dynamic***. It handles tasks like –

- Updating the UI without page reloads
- Validating user input
- Managing temporary data
- Communicating asynchronously with the server (AJAX).
- It also allows integration with ***third-party services*** and ***API***s, improving user experience and reducing server load.

Year - 2023

Section-A

1. a) Explain the roles of HTML, CSS and pure JavaScript as frontend website development technologies.

Solution:

HTML – Structure

Defines the ***skeleton*** or structure of a web page. It tells the browser *what* elements exist: headings, paragraphs, links, images, forms, tables, etc. Without HTML, there would be no content or structure to display.

Example:

```
<h1>Welcome</h1>
<p>This is a paragraph.</p>
<img src="logo.png" alt="Company Logo">
```

CSS – Presentation

Controls the ***look and feel*** of the HTML structure. Defines how elements are styled: colors, fonts, sizes, spacing, layouts, animations.

Enables ***responsive design***, making pages adjust to different screen sizes (desktop, tablet, mobile).

Example:

```
h1 {
      color: blue;
      font-size: 28px;
      text-align: center;
}
```

Without CSS, every web page would look plain, black text on a white background.

JavaScript – Behavior

Adds ***interactivity and dynamic functionality***. Makes web pages respond to user actions without reloading the entire page.

Examples:

- Dynamic content updates (e.g., live search, chat apps).
- Handling events like clicks, typing, mouse movements.
- Communicating with back-end servers via APIs (AJAX, Fetch, etc.).

Example:

```
document.getElementById("btn").addEventListener("click",   function() {
      alert("Button clicked!");
});
```

Without JavaScript, websites would be static and non-interactive.

b) What are the differences between client-side JavaScript and server-side JavaScript?

Solution:

Feature	***Client-Side JavaScript***	***Server-Side JavaScript (Node.js)***
Execution	Runs in the user's browser	Runs on the web server
Purpose	Handles UI, interactivity, form validation	Handles server logic, database access, API calls
Access	Can manipulate the DOM	Cannot access browser DOM directly
Performance	Reduces server load by processing on client	Handles multiple requests, heavy processing
Security	Vulnerable to XSS attacks	Vulnerable to server attacks, needs input sanitization

c) Name few front-end and back-end website development technologies. Explain what connects these two aspects of website.

Solution:

Front-End Technologies:

HTML, CSS, JavaScript, React, Angular

Back-End Technologies:

Python (Django, Flask), PHP, Node.js, Ruby on Rails, SQL/NoSQL databases

Connection:

The front-end and back-end communicate via HTTP requests/responses or APIs. The front-end sends user actions to the back-end, which processes data and returns response to the front-end to display dynamically.

2. Suppose you are tasked with creating a news website for a leading print newspaper in Bangladesh. The goal is to deploy a significant web application within 6 months, aiming to enhance the newspaper's digital presence, reach a broader audience, and boost overall revenue through subscriptions, advertisements, and sponsored content. To achieve this, you have adopted an incremental development approach, delivering the web application in four increments. Each increment-will focus on delivering specific features while maintaining flexibility for feedback and improvement. After several discussions with stakeholders, you have identified a set of core requirements that align with the newspaper's objectives. You are tasked with breaking these functionalities into increments. scheduling the project timeline, and defining the key user categories that will interact with the website.

a) Use your imagination to come up with a set of functionalities, and then break down these functionalities into each increment.

Solution:

Incremental Breakdown of Functionalities for News Website

Increment 1 (Month 1–2): Core News Display

- ***Basic news articles:*** Display recent news articles, organized by categories (national, international, business, sports, etc.).
- ***Search functionality:*** Enable users to search for specific articles or topics.
- ***Social media integration:*** Allow users to share news on platforms like Facebook, Twitter, etc.

Increment 2 (Month 2–3): User Engagement

- ***User accounts:*** Allow users to create accounts, save preferred sources and topics.
- ***Article commenting:*** Enable users to comment and engage in discussions.
- ***News alerts:*** Send notifications about important breaking news stories.

Increment 3 (Month 3–4): Accessibility & Contribution

- ***Multilingual support:*** Enable reading news in multiple languages.
- ***Audio news:*** Allow users to listen to news articles.
- ***User-generated content:*** Allow submission of articles or photos for publication consideration.

Increment 4 (Month 5–6): Personalization & Multimedia

- ***Personalized news feed:*** Customize news feed based on user preferences.
- ***News archive:*** Access historical news articles.
- ***Video news:*** Watch videos of news events and expert interviews.

b) Create a macroscopic schedule for the web app based on the increments you have planned

Solution:

Increments	***Key Features***	***Timeline (Weeks)***	***Deployment***
Increment 1: Core News Platform	Responsive website layout- News categories (Politics, Sports, Entertainment, Business, etc.)- CMS for journalists/editors to publish news- User accounts (basic)	Week 1 – 6	End of Week 6
Increment 2: Engagement & Multimedia	Comment system- Photo & video gallery support- Social media sharing- Newsletter subscription	Week 7 – 12	End of Week 12
Increment 3: Monetization & Personalization	Digital subscription system (paid plans)- Advertisements integration- Personalized news feed- User profile management	Week 13 – 18	End of Week 18
Increment 4: Advanced Analytics & Mobile Optimization	Real-time analytics dashboard for admins- Sponsored content management- Progressive Web App (PWA) for mobile- Performance & SEO optimization	Week 19 – 24	End of Week 24

c) Estimate your development budget based on the time and people needed to deliver the web application.

Solution:

Role	***Responsibilities***	***Monthly Cost***	***Duration***	***Total Cost***
Project Manager (1)	Planning, tracking, stakeholder communication	150,000	6 months	900,000
UI/UX Designer (1)	Wireframes, user flows, visual design	100,000	2 months	200,000
Frontend Developers (2)	React/HTML/CSS, responsive design, PWA features	120,000 each	6 months	1,440,000
Backend Developers (2)	CMS, authentication, subscriptions, analytics, APIs	130,000 each	6 months	1,560,000
QA Engineer (1)	Test planning, bug tracking, regression testing	90,000	4 months	360,000
DevOps/Cloud Engineer (1)	Server setup, CI/CD, security, monitoring	120,000	3 months	360,000
Content Migration Assistants (2)	Migrate existing articles to new platform	50,000 each	2 months	200,000

3. a) What do you understand by 'scalability" in the context of web application?

Solution:

Scalability

Scalability is the ability of a web application/system to handle increasing traffic or data by adding more resources without affecting user experience.

Need for Scalability

1. ***Serving more users*** – Increased concurrency means more connections, threads, and server load.

2. ***Processing more data*** – Large datasets can overload memory, slow computations, and increase database read/write latency.
3. ***Handling high interaction rates*** – Apps like multiplayer games or stock trading require minimal latency and fast server response.

b) How does load balancer work?

Solution:

Load Balancer Working Process

1. Client sends request to load balancer.
2. LB selects a server using an algorithm. This is done based on capacity, response time, active connections, geographic location, etc.
3. Request goes to chosen server.
4. Server processes request, sends response back to the load balancer.
5. Load balancer forwards response to the client.

c) Discuss Weighted Round Robin load balancing algorithm with its pros and cons.

Solution:

Weighted Round Robin Algorithm for Load Balancing

It is an enhanced version of round-robin load balancing algorithm that assigns weights to servers based on capacity/performance.

More powerful servers handle more requests; weaker servers handle fewer.

Pros:

- Load matches server capacity → better resource use.
- Flexible → can adjust weights for new/changed servers.
- Improves overall performance, prevents overload on weaker servers.

Cons:

- Weight assignment can be tricky → need accurate server metrics.
- Managing weights adds overhead.
- Not ideal for highly variable loads → doesn’t track real-time server load.

4. a) File upload features are a common and convenient functionality in web applications, but they can also introduce significant security risks if not properly managed. Attackers often exploit vulnerabilities in this feature to gain unauthorized access or execute malicious actions. Explain how an attacker can exploit a file upload vulnerability to execute arbitrary code on the server and how can we protect against this vulnerability.

Solution:

Performed: Attacker uploads a malicious file (e.g., PHP/ASP web shell) to the server, then executes it via direct URL to gain control.

Protection: Store uploads outside web root, validate MIME type and content, rename files, restrict executable permissions, use antivirus scanning.

b) Websites handle a vast amount of sensitive data, from user credentials to business-critical information. Without proper safeguards, unintended information leaks can occur, compromising security and user trust. What are some common ways a website might unintentionally leak information, and what measures can be taken to prevent such leaks?

Solution:

Name	***How leaks info***	***Prevention***
SQL Injection	Attackers manipulate database queries to access confidential data.	Use parameterized queries, validate and sanitize input.
Code Injection	Malicious code executed on the server can reveal sensitive files or system info.	Validate user input, disable unnecessary code execution, use secure frameworks.
File Upload Vulnerabilities	Malicious files can execute or expose server data.	Store uploads outside web root, validate file type/content, restrict permissions
Cross-site Scripting (XSS)	Injected scripts can steal cookies, session tokens, or user data.	Escape/encode user inputs, implement Content Security Policy.
Cross-site Request Forgery (CSRF)	Tricks authenticated users into performing unwanted actions, exposing data.	Use CSRF tokens, validate referer headers, enforce same-site cookies.

Section -B

5. a) How does SQL injection work?

Solution:

An attack where a hacker injects SQL control characters or clauses into inputs like forms, the application interpolates into database queries, causing the database to execute unintended SQL commands such as read, modify, delete, or escalate privileges.

Server code builds SQL by concatenating user input into query strings. Attacker supplies input containing SQL meta-characters (e.g. `'`, `--`, `;`, `OR 1=1`) that change query logic — e.g., bypass authentication or append destructive statements.

b) List some methods through which SQL statements are injected into vulnerable systems.

Solution:

Some methods by which SQL statements are injected into vulnerable systems.

1. ***Input fields:*** Login forms, search boxes (`' OR 1=1 --`).
2. ***URL query parameters:*** `id=1; DROP TABLE users;--`.
3. ***Cookies/headers:*** Injecting SQL payloads in cookie/session values.
4. ***Hidden form fields / POST data:*** Manipulating values before submission.
5. ***Second-order injection:*** Malicious input stored in DB gets executed later when reused in a query.

c) What are the traits in Laravel?

Solution:

Trait

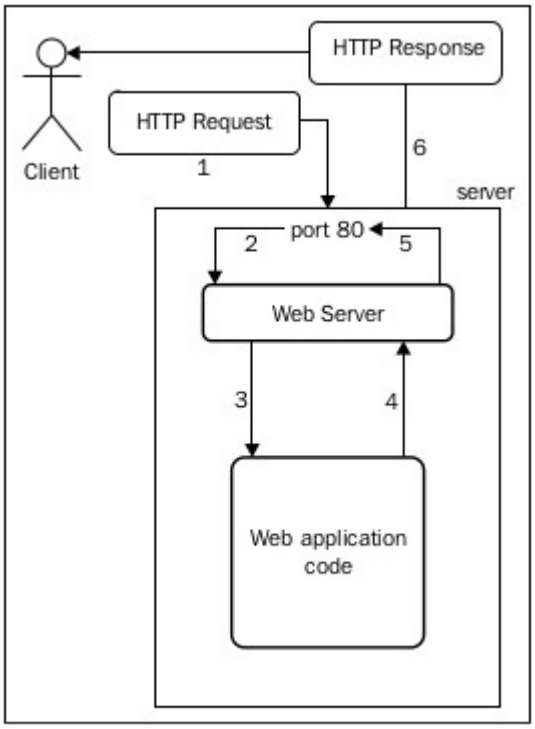
In Laravel, a ***trait*** is a mechanism for reusing code across multiple classes. Traits can contain methods and properties, but unlike classes, they ***cannot be instantiated*** on their own. They are "injected" into classes using the `use` keyword, allowing different classes to share the same functionality without requiring inheritance.

Traits in Laravel are widely used to ***avoid code duplication***. For example, you might have a trait that handles ***API responses***, ***logging***, or ***UUID generation***, and you can include it in multiple models, controllers, or services.

6. a) In web applications, the flow of data between the client and server is controlled by HTTP requests. This process involves several stages, beginning with the client clicking a hyperlink and ending with the web page being displayed in response. Can you describe the life cycle of an HTTP request from start to finish?

Solution:

1. The client (browser) sends an HTTP request (GET or POST) to the server.
2. The server listens on a specific port (e.g., 80/443) and captures the request.
3. The web server determines which web application or resource should handle the request, based on the URL path or domain.
4. The web application processes the request (business logic, database queries, etc.) and generates a response.
5. The web server sends the generated response (usually HTML, JSON, or files) back to the client.
6. The browser displays the response to the user, completing the HTTP request-response cycle.



Request-response flow on the server side

b) How can caching improve the performance of a web application?

Solution:

Caching improves web application performance by storing frequently accessed data in a fast, easily retrievable form in-memory, reducing the need to repeatedly query the database.

Benefits:

1. ***Faster response times*** – Data is retrieved from memory instead of slower disk-based storage.
2. ***Reduced server load*** – Fewer database queries lower processing overhead.
3. ***Scalability*** – Distributed caches like Redis allow multiple servers to share data efficiently, supporting high traffic.

c) Although HTTP is a stateless protocol, how does a web server maintain state and remember a client between requests during communication over the HTTP protocol

Solution:

A web server remembers a client between requests by maintaining an HTTP session. Since HTTP is a stateless protocol, the server needs a way to identify which requests belong to which client.
The most common method is through cookies:

- When the server first responds, it includes a Set-Cookie header.
- The client's browser stores this cookie, which contains a small piece of data unique to that session.
- For every subsequent request, the browser automatically sends the cookie back in the Cookie header.

This back-and-forth exchange allows the server to recognize the client and maintain a stateful conversation until the session ends.

7. a) Is there any way to automate the process of loading classes in PHP, so you don't have to manually call 'require_once' or 'include_once for each class? Explain.

Solution:

Yes, PHP provides a way to automatically load class files without manually writing `require_once` or `include_once` for each class. This is called ***autoloading***.

- When PHP encounters a ***class that hasn't been loaded yet***, it calls an ***autoloader function*** to find and include the class file automatically.
- This way, you can use the class immediately without manually including its file.

Ways to implement autoloading:

1. ***__autoload() - depricated***
    - Define a function named `__autoload($classname)` that includes the class file.
    - Limitation: ***only one autoload function*** can exist.

```
function __autoload($classname) {
       include __DIR__ . '/' . $classname . '.php';
}
```

2. ***spl_autoload_register() - recommended***
    - Register ***one or more autoload functions***. PHP tries them in order until the class is loaded.
    - Supports ***functions, static methods, or closures***, making it flexible.

```
spl_autoload_register(function($classname) {
       include __DIR__ . '/' . $classname . '.php';
});
```

3. ***Composer (PSR-4) autoloading***
    - Modern projects usually use Composer.
    - Defines namespaces and folder structure, generating an autoloader automatically.

```
require 'vendor/autoload.php';
$user = new App\Models\User(); // automatically loads the correct file
```

b) What is the function of a "Router in MVC bases web application?

Solution:

In an MVC-based web application, the router acts as the ***traffic controller***.

Its main functions are:

1. ***Intercepts incoming requests****:* It examines the URL/path of the HTTP request.
2. ***Matches the request to a route****:* It checks a predefined list of routes to determine which controller and method should handle the request.
3. ***Passes parameters to the controller****:* If the URL contains dynamic parts (like `/books/5`), it extracts them and sends them to the controller.
4. ***Delegates processing****:* It invokes the appropriate **controller action**, letting the controller handle business logic and generate a response.

c) Briefly explain the PSR-4 autoloading standard for PHP.

Solution:

PSR-4 Autoloading Standard (PHP)

Provides a ***standard way to map namespaces to file paths*** in PHP, allowing classes to be autoloaded automatically.

How it works:

- Each ***namespace corresponds to a directory***.
- Class names correspond to ***file names***.
- When a class is used, the autoloader converts the namespace into a file path and includes the file.

Example:

```
"autoload": {
      "psr-4": {
            "App\\": "src/"
      }
}
```

Using this setup:

```
use App\Models\User;
$user = new User(); // loads src/Models/User.php automatically
```

8. a) What are the characteristics of a bad website design and how to avoid them?

Solution:

Characteristics of a Bad Website Design

1. Slow loading times that frustrate users.
2. Cluttered or confusing layouts, making it hard to find information.
3. Poor color contrast or hard-to-read fonts.
4. Inconsistent navigation and structure across pages.
5. Non-responsive design, not optimized for mobile or tablet devices.
6. Excessive pop-ups, ads, or distractions that interrupt user experience.

Ways to Avoid Bad Website Design

1. Keep layouts simple, clear, and intuitive.
2. Maintain consistent navigation and structure.
3. Optimize the website for fast loading and performance.
4. Use readable fonts and good color contrast.
5. Ensure the site is responsive and works on all devices.
6. Minimize pop-ups and distractions, focusing on content quality.

b) Distinguish between performance testing, load testing and stress testing.

Solution:

Aspect	***Performance Testing***	***Load Testing***	***Stress Testing***
Purpose	Checks app's speed, response time, and availability under normal conditions.	Checks system behavior under a specific, expected load (normal & peak).	Pushes the system beyond its limits to find breaking point.
How	Simulate many users and measure response times.	Define fixed user levels (e.g., 500/1000 users) and test system response.	Overload the system with extreme traffic until failure.
Why	Users expect fast response; ensures good user experience.	Ensures the app can handle everyday and peak usage.	Ensures graceful failure and recovery under extreme load.
Failures Caused By	Inefficient code, poor resource allocation, bad deployment.	Server configuration or resource limits.	Bottlenecks in environment/resources.
Example	Amazon search results should load within 2s.	University portal must support 500 logins at once.	Test 10,000 logins at once to see if system crashes.

c) What is JSON? Briefly discuss JSON syntax.

Solution:

JSON

JSON or JavaScript Object Notation is a lightweight data-interchange format that is easy for humans to read and write, and easy for machines to parse and generate. It is commonly used to transmit data between a server and a web application.

JSON Syntax:

- ***Data is in key/value pairs:*** Each piece of data is written as `"key": value`.
- ***Objects are enclosed in curly braces { }:*** Represents a collection of key/value pairs.
- ***Arrays are enclosed in square brackets [ ]:*** Represents an ordered list of values.

- ***Values can be:*** Strings (in double quotes), numbers, objects, arrays, `true`, `false`, or `null`.
- ***Keys must be strings***, and each key/value pair is separated by a comma.

Example:

```
{
  "name": "Alice",
  "age": 25,
  "hobbies": ["reading", "cycling", "music"],
 "isStudent": false
}
```

Analysis of Question

Topics	*Questions*
1. **Web Applications Basics**	
• Definition of Web Apps, differences from traditional software • Categories & types of web-based systems • Unique characteristics of web apps • Semantic Web • Scalability & load balancing • Website design principles	• 2020: Q1 {a, b} • 2021: Q1 {a} • 2022: Q1 {a, b, c} • 2023: Q1 {a, b, c}, Q3 {a, b, c}, Q8 {a}
2. **Web Engineering & Process Framework**	
• Definition & importance of web engineering • Generic activities in web engineering • Incremental development planning & scheduling • Agility principles in web engineering teams • Risk management & handling changes • Communication with stakeholders	• 2020: Q2 {a, b, c} • 2021: Q1 {b}, Q2 {a} • 2022: Q2 {a, b}, Q3 {a, b, c}, Q4 {a} • 2023: Q2 {a, b, c}
3. **Modeling & UML / Requirements Analysis**	
• Use case diagrams & scenarios • UML sequence diagrams • Modeling importance	• 2021: Q2 {b} • 2022: Q2 {c}, Q4 {b, c}

4. **HTTP & Client–Server Communication**	
• Life cycle of an HTTP request • Maintaining state (cookies, sessions) • Caching for performance • GET vs POST • REST APIs	• 2020: Q3 {a, b} • 2021: Q8 {b} • 2022: Q8 {b} • 2023: Q6 {a, b, c}
5. **Frontend Development**	
• HTML, CSS, JavaScript roles • Client-side vs server-side scripting • Frontend vs backend technologies • Client-side scripting role • JSON	• 2020: Q3 {c} • 2022: Q8 {a, c} • 2023: Q1 {a, b, c}, Q8 {c}
6. **PHP & OOP Concepts**	
• Benefits of OOP in web apps • Classes, setters/getters, traits, inheritance, interfaces • Autoloading (spl_autoload_register, PSR-4) • require vs require_once • Anonymous functions	• 2020: Q4 {a, b, c} • 2021: Q4 {a, b, c}, Q7 {b} • 2022: Q7 {c} • 2023: Q5 {c}, Q7 {a, c}
7. **MVC Architecture & Laravel Framework**	
• MVC design pattern & its components • Router–controller interaction • ORM in Laravel • Laravel directories, artisan commands,	• 2020: Q6 {c} • 2021: Q3 {b, c}, Q6 {a, b, c}, Q8 {a, c} • 2022: Q5 {a, b, c} • 2023: Q7 {b}

migrations • Named routes in Laravel	
8. **Database & PDO / Composer Tools**	
• MySQL vs MySQLi vs PDO • PDO methods (query, exec, prepare statements) • Database design timing & planning • Composer & composer.json structure • Practical database queries & search form implementation	• 2020: Q6 {a, b}, Q7 {a, b, c} • 2021: Q3 {a}, Q7 {a, c} • 2022: Q7 {a, b}
9. **Testing & Debugging**	
• Importance of testing, AEIOU principles • Types of tests (unit, integration, acceptance, system) • Code coverage (statement/branch)	• 2020: Q5 {a, b, c} • 2021: Q5 {a, b, c} • 2023: Q8 {b}
10. **Security**	
• SQL Injection & protection • File upload vulnerabilities • Cross-site request forgery (CSRF) • General website security measures • Information leakage prevention • Cookies in PHP	• 2020: Q8 {a(i, ii, iii), b} • 2022: Q6 {a, b}, Q7 {c} • 2023: Q4 {a, b}, Q5 {a, b}