Review 2

Tourism Management System

Risk Management Process:

1.Risk: Technical Challenges

Description: Technical complexities in integrating PHP, MySQL, HTML, CSS, and other technologies may cause delays in development.
Response:
- Allocate additional time for technical research and testing.
- Include skilled technical experts in the team.
- Consider modular development to simplify integration.

2. Risk: User Acceptance

Description: End-users may find the system non-user-friendly, leading to low acceptance.
Response:
- Involve end-users in design and testing phases.
- Provide comprehensive user training and support.
- Establish iterative feedback loops for continuous improvements.

3. Risk: Insufficient Testing

Description: Insufficient testing may result in undetected bugs affecting system performance.
Response:
- Develop a detailed testing plan covering unit, integration, and user acceptance testing.
- Implement continuous testing and debugging practices throughout development.
- Conduct thorough quality assurance to ensure a robust system.

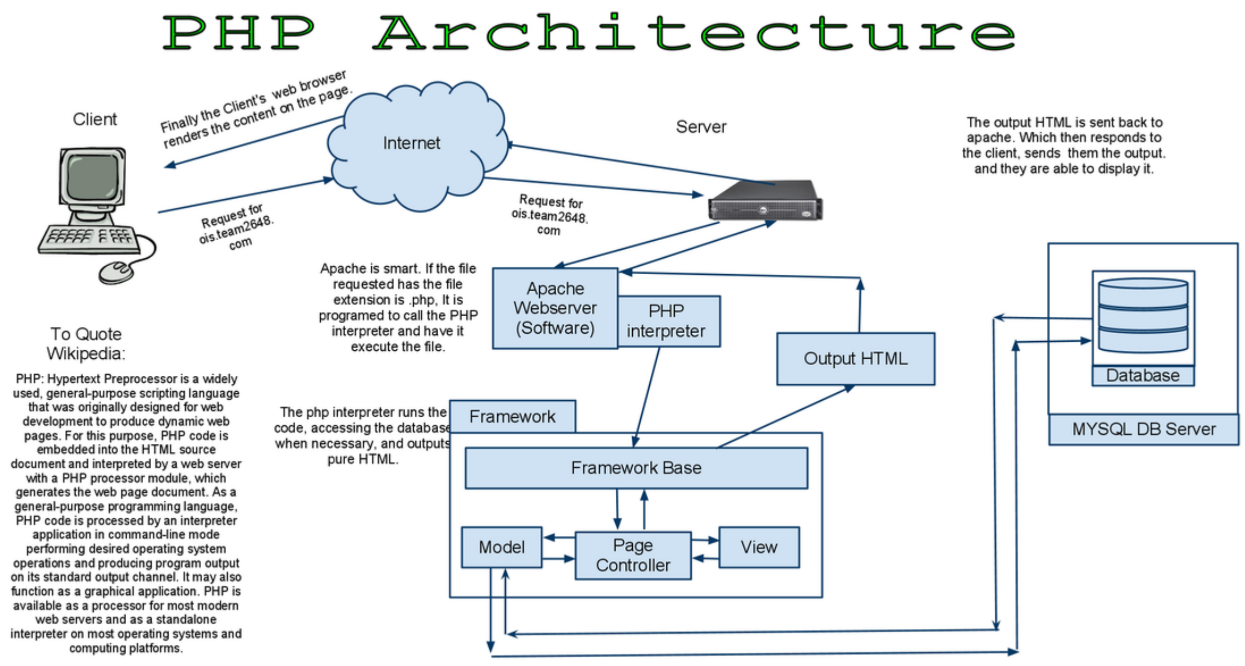
Contingency Plans:

- Establish a rapid response team to address critical issues promptly.
- Maintain open communication channels with teammates for immediate issue reporting.
- Document and analyze past project risks to enhance future risk management strategies.

Ongoing Monitoring:

- Regularly review and update the risk register.
- Hold periodic risk assessment meetings with the project team.
- Stay vigilant for emerging risks and adapt strategies accordingly.

Architecture of Front End user:



Finally the Client's web browser renders the content on the page.

The output HTML is sent back to apache. Which then responds to the client, sends them the output. and they are able to display it.

Apache is smart. If the file requested has the file extension is .php, It is programed to call the PHP interpreter and have it execute the file.

The php interpreter runs the code, accessing the databas when necessary, and outputs pure HTML.

PHP: Hypertext Preprocessor is a widely used, general-purpose scripting language that was originally designed for web development to produce dynamic web pages. For this purpose, PHP code is embedded into the HTML source document and interpreted by a web server with a PHP processor module, which generates the web page document. As a general-purpose programming language, PHP code is processed by an interpreter application in command-line mode performing desired operating system operations and producing program output on its standard output channel. It may also function as a graphical application. PHP is available as a processor for most modern web servers and as a standalone interpreter on most operating systems and computing platforms.

source: - http://www.techwizworld.net/blog/php-architecture

The query cache plugin in PHP, implemented as a C-based extension, operates within PHP by registering as a mysqlnd plugin during interpreter startup. This integration allows it to modify the behavior of PHP MySQL extensions (mysqli, PDO_MYSQL, MySQL) using the mysqlnd library without altering the extension's API. This seamless integration ensures compatibility with all PHP MySQL applications while enabling transparent usage.

Transparent to Use:

At runtime, PECL/mysqlnd_qc can proxy queries sent from PHP to the MySQL server. It inspects SQL statements to determine whether to cache results. Cached data is stored using various storage handlers, and subsequent executions of the statement are served from the cache within a user-defined period. The caching decision is made either globally or per statement basis using SQL hints (/*qc=on*/).

Flexible Storage:

The plugin supports different storage handlers, allowing diverse cache entry scopes. These scopes range from process memory (default) to shared memory (APC), SQLite, and MEMCACHE. Switching between handlers is possible at runtime, but a consistent choice is recommended to avoid complexity.

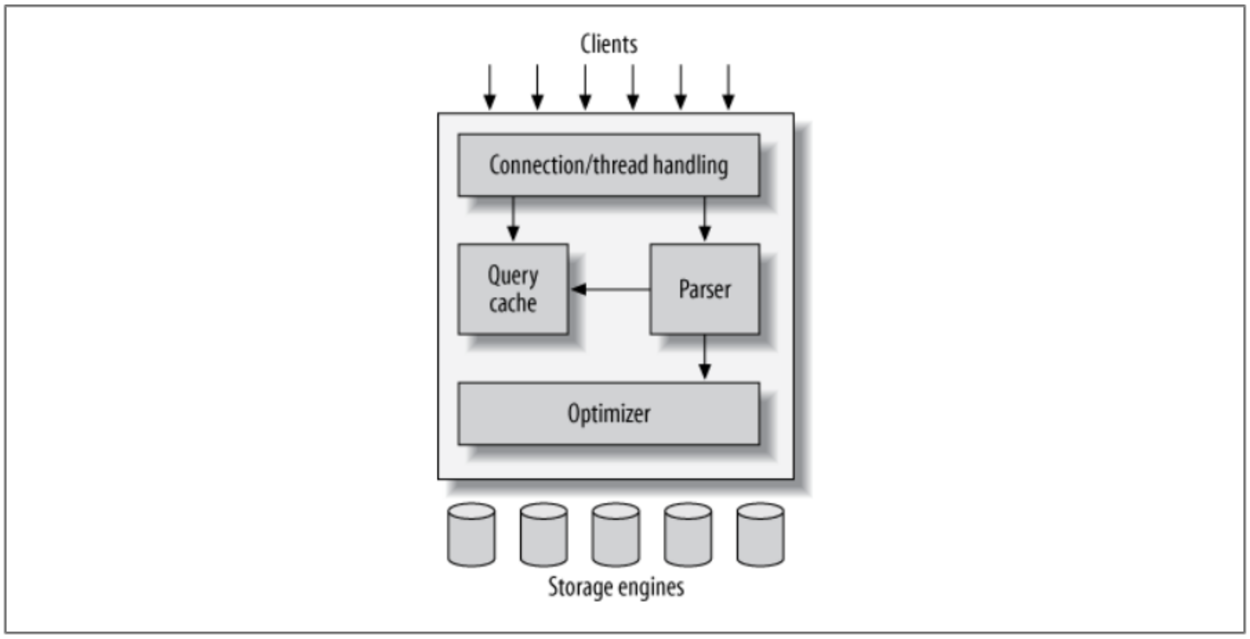
Built-in Slam Defense to Avoid Overloading:

PECL/mysqlnd_qc includes a built-in slam defense mechanism to prevent overloads. When a popular cache entry expires, the plugin extends its lifetime briefly, allowing refreshing clients to access it without triggering concurrent refresh attempts. This approach prevents overload situations by managing concurrent accesses efficiently.

Unique Approach to Caching:

PECL/mysqlnd_qc employs a unique caching approach superior to application-based solutions. Instead of serializing PHP variables for storage, it directly stores raw wire protocol data sent from MySQL to PHP. This method eliminates the need for extra serialization steps, enhancing efficiency and improving cache performance.

MySQL's Logical Architecture: -



source: - https://shashwat-creator.medium.com/mysqls-logical-architecture-1-eaaa1f63ec2f

The MySQL architecture consists of layers. The top layer includes essential services like connection handling, authentication, and security, common to most network-based tools. The third layer houses storage engines responsible for storing and retrieving data. Each engine, like filesystems in GNU/Linux, has unique features.

The server communicates with these engines through the storage engine API, which abstracts differences between engines, making them transparent at the query layer. These engines respond to server requests but do not parse SQL or communicate with each other.

Components:

1. **User Interface Components:**
   Web pages, forms, and interactive elements are designed using HTML, CSS, and JavaScript.
2. **frontend Components:**
   PHP scripts responsible for processing user requests, implementing business logic, and interfacing with the database.
3. **backend Components:**
   MySQL database schema storing user profiles, booking information, feedback, and other relevant data.

Interfaces:

1. **User Interface to Application Tier:**
   User interactions from the interface are processed by PHP scripts, which validate user inputs and invoke appropriate methods.
2. **Application Tier to Data Tier:**
   PHP scripts communicate with the MySQL database using SQL queries to retrieve, update, and store data.
3. **External Services:**
   Integration with external services (if any) such as payment gateways or mapping APIs is achieved via secure HTTPS protocols, ensuring data integrity and user security.

Design Pattern:

The chosen design pattern is the **Model-View-Controller (MVC)** pattern. MVC separates the application logic into three interconnected components:

1. **Model:** Represents the application's data and business logic. In this system, it corresponds to the database schema and PHP scripts handling data processing.
2. **View:** Displays the data to users and handles user input. It encompasses the HTML, CSS, and JavaScript components for the user interface.
3. **Controller:** Manages user input, processes it, and interacts with both the Model and View. PHP scripts act as controllers, managing the flow of data between the Model and View.

Reason for Choosing MVC:

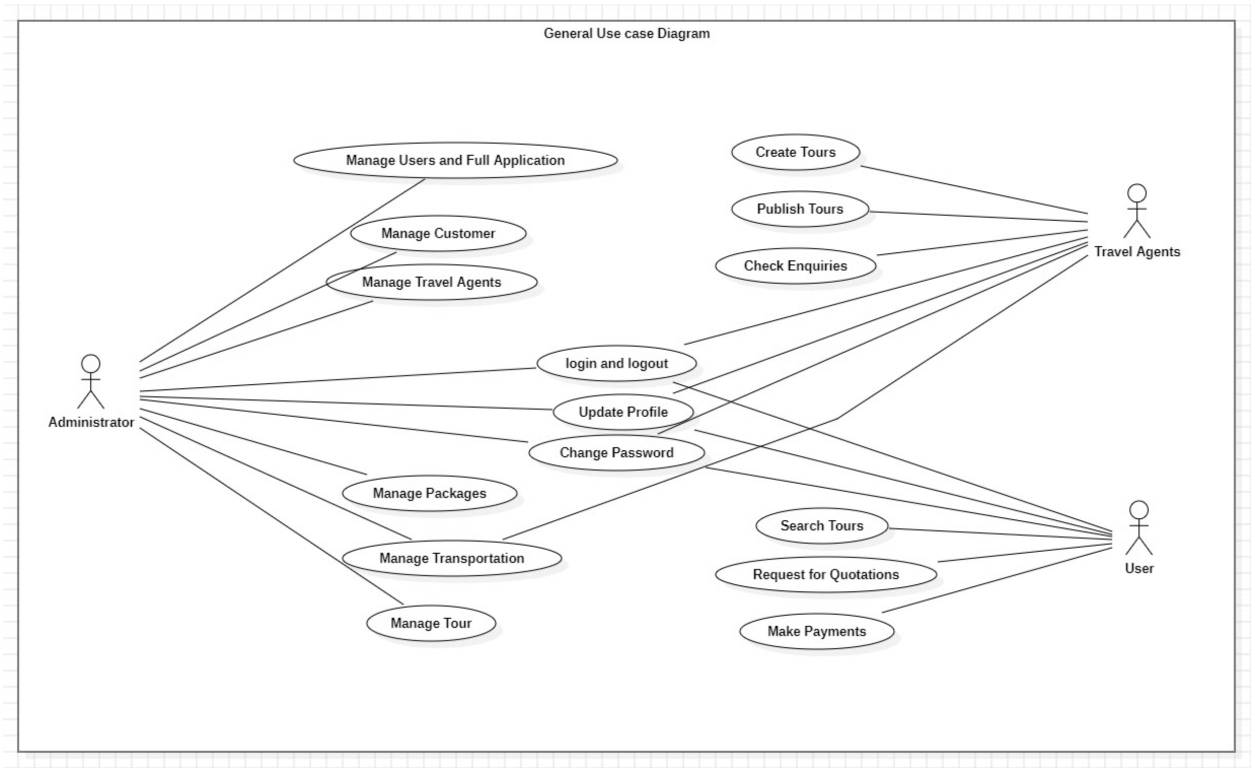
MVC promotes modularity, maintainability, and reusability in software development. By separating concerns, it enhances code organization and readability. In the context of the Travel and Tourism Management System, MVC ensures a clear distinction between user interface design, data processing, and user interactions, making the system scalable and easier to maintain. Additionally, MVC aligns with best practices, allowing for a streamlined and efficient development process.

Performance and Scalability:

1. **Caching**: Implement caching mechanisms to reduce database queries and enhance performance. This can be achieved using technologies like Memcached or Redis.
2. **Database Optimization**: Proper indexing, normalization, and database maintenance routines will be employed to ensure optimal database performance.
3. **Load Balancing**: If the application experiences high traffic, we'll use load balancing to distribute the load across multiple servers, improving scalability.

By adhering to the MVC pattern and considering performance and scalability, we aim to create a robust and efficient Tourism Management System. This design will allow for future enhancements and updates with minimal disruption to the existing system.

UML diagram: -



Review the calendar activities: -

How is the work you proposed on the first report going?

we are diligently following our planned course of action. We have successfully made progress in implementing a portion of the frontend. Our website now includes multiple interactive pages. Specifically, our focus has been on developing the homepage, where significant achievements have been made. We have successfully implemented various user interface components using HTML and CSS. Thus far, we have not encountered any issues, and our project plan remains unchanged. In the event that challenges arise, we are prepared to address them promptly to ensure the project's smooth continuation. It is worth noting that we anticipate a relatively straightforward development process as our website primarily caters to travel agents and customers.

Advance of the project in terms of the objectives: -

The project has made progress in implementing a portion of the frontend. Our website has multiple interactive pages. Up to this point, we have been working on the homepage. Specific achievements include the development and implementation of various user interface components using HTML, CSS.

Changes or Modifications: -

The project is still in its initial stages. We are following our planned course of action, and so far, we have not encountered any issues. Therefore, we have not made any changes to the project plan. If we do encounter any problems, we will work to rectify them and continue following the planned course of action. It is possible that we may not face any issues because this is a straightforward website designed for travel agents and customers.

Possible Difficulties: -

1. **Technical Challenges:** Integrating complex technologies like PHP, MySQL, HTML, CSS, and JavaScript may pose technical difficulties, leading to delays or errors during development.
2. **User Acceptance:** Ensuring that the user interface is intuitive and user-friendly can be challenging. If users find the system difficult to navigate, it might result in low adoption rates.
3. **Testing and Debugging:** Inadequate testing might lead to undetected bugs or issues, affecting system stability and user experience. Debugging complex interactions between frontend and backend components can be time-consuming.