1. General Questions
Q: What is the purpose of your event planner website?
A: The purpose of this Event Planner website is to allow users to efficiently organize, manage, and track events. It enables users to create, edit, delete, and manage event details while allowing attendees to view or register for events.

Q: How does your system help users manage events?
A: Users can:

Log in and access their event dashboard.
Create new events by filling out a form with event details.
Edit or delete events when necessary.
View upcoming events in an organized manner.
Q: What technologies did you use to develop this project?
A: The project is built using:

Frontend: HTML, CSS, JavaScript (for UI & interactivity).
Backend: PHP (for authentication, event management).
Database: MySQL (to store user and event details).
Security: Session management, SQL injection prevention, password encryption.
Q: How is your website different from existing event management platforms?
A: Unlike general event platforms, this system is:

Customizable: Built specifically for targeted event types.
Simple & Lightweight: Focuses on core event planning features without unnecessary complexity.
Secure: Implements best security practices for authentication and data protection.
2. Frontend (HTML, CSS, JavaScript)
Q: How did you design the user interface for your website?
A: The UI was designed using HTML & CSS to create a user-friendly and intuitive experience, including:

A navigation bar for easy access.
Forms for login and event creation.
Cards & tables to display event details clearly.
Q: What CSS techniques did you use for styling?
A: I used:

Flexbox & Grid Layout for responsive designs.
CSS Transitions & Hover Effects for interactivity.
Media Queries to ensure mobile compatibility.
Q: Did you use any JavaScript for form validation or dynamic updates?

A: Yes, JavaScript was used for:

Client-side validation (ensuring required fields are filled).
Form input validation (e.g., proper email format).
Dynamic updates without page reload (e.g., live form feedback).
Q: How did you ensure that your website is responsive?
A: I used CSS media queries to adapt the design for different screen sizes and ensured that elements resize properly for mobile and tablet users.

3. Backend (PHP, MySQL, Security)
Q: How does your login system work?
A: The login system:

Accepts email and password input from the user.
Queries the MySQL database for matching credentials.
If correct, starts a session and redirects the user to their dashboard.
Q: How do you store user credentials securely?
A: I use:

password_hash() to encrypt passwords.
password_verify() to check passwords during login.
Q: Why did you use session_start() in your PHP files?
A: session_start() is used to:

Track logged-in users.
Maintain session variables across different pages.
Q: How do you prevent SQL injection in your login system?
A: I use prepared statements with mysqli to prevent SQL injection.
Example:

php
Copy
Edit
```
$stmt = $conn->prepare("SELECT * FROM users WHERE email = ?");
$stmt->bind_param("s", $email);
$stmt->execute();
```
Q: Why is md5() not recommended for password storage?
A: md5() is not secure because:

It produces predictable hashes (easy to crack).
It lacks salting, making it vulnerable to brute-force attacks.
Q: How do you handle errors if a user enters incorrect login credentials?
A: The system:

Displays an error message (Invalid username or password).
Prevents brute force attacks by adding login attempt limits.
Q: What happens if multiple users try to log in at the same time?
A: The system handles multiple logins using PHP sessions. Each user gets an individual session ID, allowing multiple users to log in without interference.

Q: How do you manage session expiration and logout functionality?
A: Sessions expire after inactivity and users can log out by calling:

php
Copy
Edit
session_destroy();
header("Location: login.php");
This clears session data and redirects the user to the login page.

4. Database & SQL
Q: How is your database structured?
A: The database consists of:

users → Stores user details (id, name, email, password).
events → Stores event details (id, event_name, date, location, user_id).
Q: What are the main tables in your database?
A: The main tables are:

users → Handles user authentication.
events → Stores event details.
Q: How do you store and retrieve event details?
A: Event details are stored in MySQL using INSERT INTO events (...) VALUES (...).
To retrieve events, we use:

php
Copy
Edit
SELECT * FROM events WHERE user_id = ?;
to fetch only events related to the logged-in user.

Q: Can you explain a SQL query you used in your project?
A: Fetching user events:

php
Copy
Edit

SELECT * FROM events WHERE user_id = ? ORDER BY event_date ASC;
This retrieves events for a specific user, sorted by date.

Q: What is the role of mysqli_real_escape_string() in your queries?
A: It escapes special characters in user input, preventing SQL injection. However, prepared statements are a more secure alternative.

Q: How would you optimize database queries for performance?
A:

Using indexes on frequently searched columns.
Limiting results with LIMIT.
Avoiding SELECT * and selecting only required columns.
5. Features & Functionality
Q: How does a user create and manage an event?
A: Users fill out a form and submit event details, which are stored in the database. They can later edit or delete their events.

Q: How do you handle event booking or RSVP functionality?
A: Users can click a "Register" button, which updates a registrations table in the database.

Q: What happens when multiple users try to book the same event?
A: The system checks availability before confirming a booking, preventing overbooking issues.

Q: Can users update or delete their events?
A: Yes, users have "Edit" and "Delete" buttons for event management.

6. Security & Performance
Q: How do you handle authentication and user sessions securely?
A:

Secure password hashing with password_hash().
Session validation to prevent session hijacking.
Q: What steps have you taken to prevent common web vulnerabilities?
A:

SQL Injection: Using prepared statements.
XSS: Escaping user inputs before displaying them.
CSRF: Implementing CSRF tokens for forms.
Q: Have you implemented any caching mechanisms?
A: Yes, server-side caching (storing frequently accessed queries) and browser caching for static assets.

7. Deployment & Future Improvements

Q: How would you deploy this website?
A: Upload files to a web hosting service, import the MySQL database, and update configurations.

Q: How would you add payment gateway integration?
A: By integrating PayPal, Stripe, or Razorpay API for secure payments.