Coding Standard

To ensure code quality, maintainability, and consistency throughout the development of the Game Hall Booking System (TurfMate), a set of standardized coding conventions and guidelines has been established. These standards define the rules and best practices for writing clean, structured, and secure code across all components of the system, including the backend (PHP and MySQL) and the frontend (HTML and CSS).
By adhering to these conventions, the system will maintain a high level of software quality, with emphasis on clarity, scalability, and security. Furthermore, consistent coding practices facilitate easier debugging, testing, and future updates or enhancements to the system.

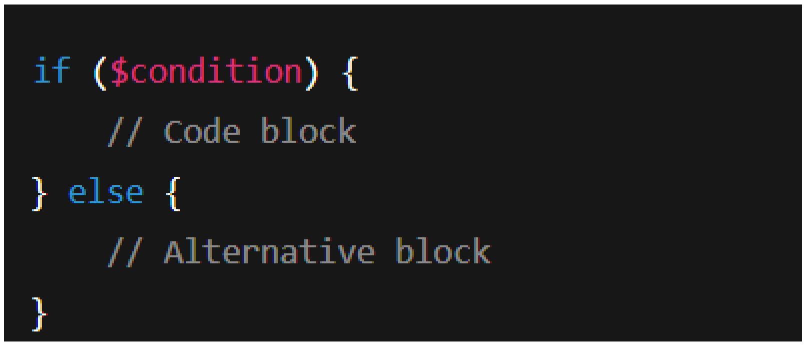
General Guidelines

- All code must be properly indented and commented for better readability.
- Each file should begin with a short header comment describing its purpose, author, and date of creation.
- Meaningful and descriptive variable, function, and file names should be used throughout the project.
- Avoid code duplication by using functions and include files wherever applicable.
- Maintain consistent naming conventions and folder structures (e.g., /includes, /css, /js, /images, /admin, /customer).

PHP Coding Standards

- **File Extension**: All PHP files use the .php extension.
- **Naming Convention**:
    - File names use lowercase with underscores (e.g., manage_products.php, add_booking.php).
    - Variables use camelCase (e.g., $bookingDate, $totalPrice).

    - Constants are written in UPPERCASE (e.g., DB_SERVER, SITE_NAME).
    - Functions use camelCase (e.g., calculateTotal(), validateUser()).
- Indentation: Use 4 spaces per indentation level (no tabs).
- Braces: Use the K&R style



```
if ($condition) {
    // Code block
} else {
    // Alternative block
}
```

Database Access:

- All database connections are handled through a central db.php file.

Error Handling:

- Use proper validation and error messages.
- Display user-friendly messages; internal errors should not be exposed to users.

Security:

- Validate and sanitize all user inputs using PHP filters.
- Use session management for user authentication.
- Store passwords using hashed encryption (password_hash() / password_verify()).

HTML Coding Standards

- Use HTML5 Doctype (<!DOCTYPE html>).
- Properly nest and close all tags.

- Use semantic HTML elements (e.g., <header>, <section>, <footer>, <nav>).
- Keep structure and content separate from styling and scripting.
- Use descriptive IDs and classes for elements (e.g., id="turf-list", class="booking-form").
- Ensure forms have proper labels, validation, and input names.

CSS Coding Standards

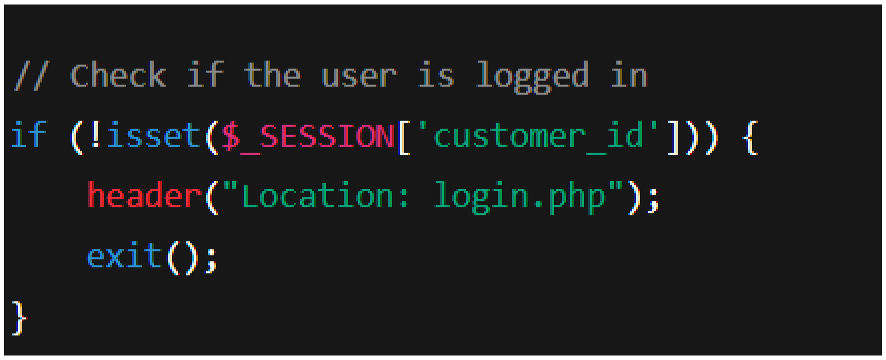
- Maintain a separate stylesheet file (e.g., style.css).
- Use class selectors instead of inline styles to ensure reusability.
- Naming conventions:
    - Class names use kebab-case (e.g., .booking-form, .nav-bar, .turf-card).
- Organize styles in logical sections (e.g., Header, Navigation, Content, Footer).
- Use comments (/* Section Name */) to separate CSS sections.
- Maintain color consistency using CSS variables or predefined color palette.
- Ensure responsiveness using media queries.

MySQL Database Standards

- Use lowercase letters with underscores for table and column names (e.g., customer_info, turf_booking).
- Define primary keys as id or *_id (e.g., customer_id, booking_id).
- Use appropriate data types (e.g., INT, VARCHAR, DATE, DECIMAL).
- Define foreign key relationships to maintain referential integrity.
- Write SQL queries in uppercase for keywords (e.g., SELECT, FROM, WHERE).
- Avoid SELECT *; specify column names explicitly for clarity and performance.

Commenting Guidelines

- Every function and file should have a short description comment explaining its purpose.
- Inline comments should describe complex logic or calculations.
- Example:



```php
// Check if the user is logged in
if (!isset($_SESSION['customer_id'])) {
    header("Location: login.php");
    exit();
}
```

File Organization

- Each PHP, HTML, CSS, and JavaScript file must be properly named to reflect its purpose (e.g., manage_turf.php, add_booking.php, style.css).
- Related files should be grouped into appropriate folders, such as:
    - /includes/ → for configuration and reusable components (e.g., db.php)
    - /admin/ → for admin dashboard and management pages
    - /customer/ → for customer interface pages
    - /assets/ → for CSS, images, and scripts

Folder Structure Convention

TurfMate

```
|
├── admin  → Admin portal pages
├── customer → Customer-facing pages
├── includes  → Reusable files (db.php, header.php, footer.php)
├──  css  → Stylesheets
├── images  → Images and icons
├── js   → JavaScript (if needed)
└── index.php  → Main landing page
```

Testing and Validation

- Validate all form inputs both on client-side (HTML5 validation) and server-side (PHP).
- Use browser tools (e.g., Chrome DevTools) to check layout and console errors.
- Use W3C HTML/CSS Validators to maintain compliance with web standards.