## Coding Style Standards:

We should have the following standards for writing our code in Medixcel:

1. Your code MUST follow [PSR-2 standards](https://www.php-fig.org/psr/psr-2/). <https://www.php-fig.org/psr/psr-2/>
2. All your classes MUST be autoloaded with [PSR-4 Standard](https://www.php-fig.org/psr/psr-4/) - <https://www.php-fig.org/psr/psr-4/>
3. All functions MUST have type hinting
4. Your variable name must be min of three characters.
5. Your variable must have prefix to define the type for example $iAge, $bRequired, $sName etc
6. Developers MUST use an indent of 4 spaces
7. PHP function files must not end with ?> & no space must be kept before <?php
8. Class names MUST be declared in StudlyCaps.
9. Method or variable names MUST be declared in camelCase.
10. Developers must not hardcode anything that can change in the future.
11. Don’t keep any commented-out code in Files. We have git for history anyway
12. Code written by developers must be reusable. The same code shouldn’t repeat at multiple places
13. Data should only be fetched when it will be displayed to the user. We shouldn’t prefetch data.
14. Whenever we display a list of data to the User then Pagination is a must unless we know that entries for that specific data won’t be more than 20.
15. Developers must make sure that no internal errors are not displayed to the user.
16. Use of `eval` & `exec` must be avoided and should only be used if permitted by the Project manager.

## File Handling:

1. File handling & access must be done using the File System interface
2. Store filename as UUID
3. File must be served to the user using File service with appropriate token type based on the access required
4. Temporary files created must be cleaned up after the usage

Database:

1. All queries must be written using DBAL with query builder or prepared query. Query builder is recommended
2. All Input parameters must be used with parameterized binding in the query. Parameter binding should be done as named parameter
3. All table names must be prefixed with “mxcel” for EHR & “cph” for CPH modules.
4. All table names must-have module name as prefix. For example, teleconsultation related tables should start with “mxcel\_teleconsultation\_meetings”
5. Each table should have the following columns:
   1. `deleted` column which represents if the data was deleted
   2. `added\_by` column which reflects who added the data
   3. `added\_on` column which reflects when the data was added
6. Each table should be indexed properly while creating or updating the table
7. Your code must not try to access data with deleted flag = 1. If any data needs to be marked as inactive then a separate flag should be used.
8. We must use null values instead of using empty string or ‘0000-00-00’ or ‘00:00’ for storing no values in date, datetime, or time type column in the database.
9. Use db connection instance as per the usage requirement as below:
   1. **db**: For reading the data
   2. **db.write**: For writing the data
   3. **db.reporting**: For generating the reports or analytics

Output Handling:

1. [Output class](https://docs.google.com/document/d/1ofOPdUNFyfy4ZE9bA7ruIKj8ybR9hg_24l1KgnCz-RU/edit) must be used to echo any content.

Input Handling:

1. Input must be sanitized using [the Input class](https://docs.google.com/document/d/15PR-_6i_eGP8UAG7M0Y_1rstLaMN6G_YMH_JqlkR0lo/edit)

CSRF Protection:

1. Each form or Ajax request must be protected using [CSRF token](https://docs.google.com/document/d/18xs_e39C_Ou9RMi5kqBFWgo1mkT6cX9Lhk4wQLI9sAw/edit).

Authentication & Authorization checks:

1. All pages must have a session check.
2. All API endpoints must be protected with API Guard or Signature-based authentication.
3. All actions on the UI must be prevalidated with authorization.
4. All authorization checks must be included on the backend side using the Permission handler class
5. Each peer reviewer needs to make sure that the new code that is being added must not put the security of the application at risk.

Audit Trail:

1. All actions must be capture in the audit trail with enough details
2. All audit trail types must be store in the [audit trail type master list](https://docs.google.com/spreadsheets/d/1uWFaUuGz7wwepWCaoaGJjodpzXb-n0Q__J-9fO9eXH0/edit#gid=0).
3. Use the hardcoded type id in the SQL query allocated on the audit trail type master list. Your SQL query must not depend on the auto increment to allocate the type id.
4. Each audit trail should be captured with a related Patient ID or Schedule ID unless the audit type is not associated with them.

Permissions:

1. Appropriate permissions must be created for the modules and for each action inside it.
2. Permissions must be checked on the UI as well as on the server-side
3. All new permissions must be captured in the [Permissions master list](https://docs.google.com/spreadsheets/d/1GSgYL-s_PK8PW6keX9VQ-gGCyc1mjPS7QwYjfHs_Vhw/edit#gid=1797239471)
4. Use the hardcoded action id, feature id in the SQL query allocated on the permission master list. Your SQL query must not depend on the auto increment to allocate the action id or feature id.

Ajax:

1. Don’t use of AjaxEhr & AjaxCph. These files are getting larger and garbage of code for all ajax calls. Instead, use Ajax Mapping with a function like in [this file](https://bitbucket.org/plus91-team/medixcel-base/src/develop/api/patient.php).
2. Each module/feature must have its own ajax file.
3. We should have **'Content-Type: application/json'** header when sending a JSON ajax response. We mustn’t use **`$.parseJSON(`** in frontend.
4. Ajax-related files must have session checks or any other type of user authentication system.

Session:

1. All session-related variables should be accessed by **SessionManager** class instead of using direct variables like $iUserID, $iStaffID, etc

Logging: