Clean Code Development (Cheat Sheet)

Definitions

- Exception Handling

EX : By using try...catch clause, exception in the code scope can be captured and handled with ease.

```
 try{
      //In Desc Order By Timestamp in seconds
      $files = scandir($dir,1);
      $filename = $files[$order];
    } catch (Exception $e){
       Log::debug($e->getMessage());
    }
```

- Abstraction (Interfaces/Classes/Functions) For Easy Readability

EX : Utilising reasonable abstraction in the code when required, further boosting the readability and maintainability of the code by hiding under-levelled definitions , thus reducing messiness.

```
  if(strpos($message,'_FILE_') !== false){
          $this -> fileUploadDialog($bot);
        }elseif(strpos($message,'_FILEOPTS_') !== false){
          $this -> askFileOptions($botman);
        }elseif(strpos($message,'_FSHOW_') !== false || strpos($message,'_FTRANSLATE_') !== false
|| strpos($message,'_FSUMMARY_') !== false ){
          $this -> handleFileOptions($message,$bot,$botman);
        }elseif(strpos($message,'_USER_') !== false){
          $this -> handleGreetingDialog($botman,$bot,$message);
        }elseif(strpos($message,'_BACK_') !== false){
          $bot->replyWithDelay("So... what do you want to talk about next");
        }
        else{
          $this -> askGPT($bot, $this->prompt_prefix.$message.$this->fixed_persona_injection);
        }
```

- Function With Clear Purpose

EX : Functions should be written to only accommodate a clear task .

```
public function askGPT($bot, $request)
   {
      if ($request){
         $response = $this -> communicateGPT4($request);
         if ($response->getStatusCode()==200){
            $bot ->
replyWithDelay($this->hex2Umlaub(json_decode($response->getBody())->Response));
         }
         else{
            Log::debug($response->getBody());
            $bot -> replyWithDelay('Sorry, it seems there is a problem at this time.');
         }
      }else{
         $bot -> replyWithDelay('Sorry, I do not understand your request.');
      }

   }
```

- Meaningful Naming Conventions And Comments (Methods/Variables/Properties)

EX : Naming of Methods,Variables,Properties,Classes and ... , also comments should be meaningful and not to confuse .

```
 public function retrieveImage($dir) {
     $top = 0;
     $latestfile = null;
     try{
        //In Desc Order By Timestamp in seconds
        $files = scandir($dir,1);
        $filename = $files[$top];
        $fullpath = $dir.$filename;
        $latestfile = new
Image(str_replace('/opt/laravelprojects/mygpttranslator/mysimpleGPTBot/storage/uploads/','htt
p://localhost:8000/file-access/',$fullpath));
     } catch (Exception $e){
        Log::debug($e->getMessage());
     }

     return $latestfile;
   }
```

- Tests With Clear Goal (Single Assert OR Multiple Asserts which are related)

EX : Tests should be written with clear objective of what would be tested within the scope.

```
/**
   * test for hex2Umlaub function.
   * @dataProvider stringProvider
   * @return void
   */
  public function testHex2Umlaub($targetedString,$expectedResult)
  {
      $controller = new BotmanController();
      $substringsToCheck = ["ä","ö","ü","ß","Ä","Ö","Ü"];
      $result = $controller->hex2Umlaub($targetedString);
      $this->assertEquals($expectedResult,
          $this->containsAnySubstring($result, $substringsToCheck),
      );
  }

  /**
   * test for retrieveImageFilename function.
   * @dataProvider pathProvider
   * @return void
   */
  public function testRetrieveImageFilename($dir,$topFilename,$expectedResult)
  {

      // Test Start
      Log::shouldReceive('debug')->andReturnNull();

      $controller = new BotmanController();

      $targetFile = $controller->retrieveImageFilename($dir);

      #Multiple asserts to check for related conditions regarding to retrieve Image filename
      $this->assertNotEmpty($targetFile);
      $this->assertFileExists($dir . '/' . $targetFile);
      $this->assertEquals($expectedResult,$this->isEqual($topFilename,$targetFile));
  }
```

Build Management & Automation

Types:

Jenkins
Maven/Gradle
Github Actions
...

Usage:

To automate application build for different environments (prod/staging), version management ,also for running tests and run code analysis to ensure healthy CI/CD .

Github Actions Setup :

Ensure .github/workflows/build.yml in the root of the project (if not, must specify in the .yml file)

Example of build.yml (Trigger for each push/pull to/from main github branch) :

```
name: Laravel CI

on:
  push:
    branches:
      - main
  pull_request:
    branches:
      - main

jobs:
  build-and-test:
    runs-on: ubuntu-latest
    defaults:
      run:
        working-directory: mysimpleGPTBot

    services:
      mysql:
        image: mysql:5.7
        env:
          MYSQL_DATABASE: botman_chatbot
          MYSQL_USER: root
          MYSQL_PASSWORD:
          MYSQL_ROOT_PASSWORD:
        ports:
          - 3306:3306

    steps:
      - name: Checkout code
        uses: actions/checkout@v2

      - name: Display working directory path
```

```
      run: |
        pwd

    - name: setup environment
      run : ../setenv.sh

    - name: Install Composer dependencies
      run: composer install --no-interaction --optimize-autoloader

    - name: Install NPM dependencies and compile assets
      run: |
        npm install
        npm run production

    - name: Create .env file
      run: cp .env.example .env
      env:
        DB_CONNECTION: mysql
        DB_HOST: 127.0.0.1
        DB_PORT: 3306
        DB_DATABASE: botman_chatbot
        DB_USERNAME: root
        DB_PASSWORD:

    - name: Generate application key
      run: php artisan key:generate

    - name: Run tests
      run: php artisan test --coverage-clover=tests/coverage/coverage.xml
--log-junit=tests/coverage/test-results.xml

    - name: Optimize Laravel for production
      run: php artisan optimize

    - name: Clear caches
      run: |
        php artisan cache:clear
        php artisan config:clear
        php artisan route:clear
        php artisan view:clear

    # You can add more steps here if needed, such as restarting your web server
    #
    - name: Build process completed successfully
      run: echo "Build process completed successfully."
```

```
  #Optional : Enable SonarScan to analyze code. Make sure sonar server + github is properly setup before enabling.
  #sonarqube:
  #  needs: build-and-test
  #  runs-on: ubuntu-latest
  #  steps:

  #  - uses: actions/checkout@v3
  #    with:
  #      # Disabling shallow clones is recommended for improving the relevancy of reporting
  #      fetch-depth: 0

  #  - name: SonarQube Scan
  #    uses: sonarsource/sonarqube-scan-action@master
  #    env:
  #      SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
  #      SONAR_HOST_URL: ${{ secrets.SONAR_HOST_URL }}
```