ステップ	実装すること	新規追加したファイル	変更したファイル	削除したファイル
step1	ルーティングを実装し、index.phpからコントローラクラスへの疎通を確認します。	* app/Modules/User/Controllers/UserController.php * app/Modules/User/Controllers/IndexController.php * public/index.php * public/.htaccess * public/composer.json		
step2	DIコンテナ(pimple)に、メール送信クラスとロガーを登録します。	* app/Libs/Mailer/*.php (メール送信のスケルトンクラス)	* public/composer.json (pimple,monolog,swiftmailerを追加) * public/index.php (DIコンテナへの登録処理を追加)	
step3	他のクラスからも、DIコンテナに入っているメーラーとロガーを取得できるようにします。	* app/Libs/Core/Container.php (pimpleコンテナのラッパークラス) * app/Libs/Core/Traits/SingletonTrait.php	* public/index.php (DIコンテナへの登録処理を変更) * app/Modules/User/Controllers/UserController.php (DIコンテナからの取得処理を追加)	
step4	キャッチできなかった例外がindex.phpまでスローされたときのために、共通の例外処理を実装します。	* app/Modules/Common/Controllers/ExceptionController.php * app/Libs/Core/Exception/*Exception.php	* public/index.php (例外処理を追加)	
step5	index.phpをスリムにします。	* app/Core/container.php * app/Core/handle-exception.php * app/Core/routes.php	* public/index.php (ファイルを分けてスリムにした)	
step6	Viewクラスを実装し、コントローラーとビューを分離します。	* app/Libs/Core/View.php * app/Helpers/*.php * app/Modules/User/Layouts/layout*html * app/Modules/User/Views/user/index.html * app/Modules/Common/Views/exception/index.html	* app/Modules/User/Controllers/UserController.php (ビューを分離)	
step7	モデルクラスとリポジトリクラスを実装し、データベースに接続します。	* app/Libs/DataSource/PdoConnector.php * app/Libs/DataSource/DataSourceConnectorInterface.php * app/Libs/AbstractSqlRepository.php * app/Libs/AbstractMariadbRepository.php * app/Models/TestModel.php * app/Repositories/TestRepositoryInterface.php * app/Repositories/TestMariadbRepository.php	* app/Modules/User/Controllers/UserController.php (モデルを呼び出す処理を追加)	
step8	設定ファイルを用意し、データベース接続情報などをハードコードしないようにします。	* app/Config/application.ini * app/Libs/IniFileParser.php * app/Libs/ApplicationConfig.php	* app/Libs/DataSource/PdoConnector.php (DSNをハードコードしないようにした)	
step9	HTTPリクエスト、アップロードファイル、セッションに関するクラスを作り、動作確認をします。	* app/Libs/Core/Request.php * app/Libs/Core/RequestInterface.php * app/Libs/Core/Session.php * app/Libs/UserSession.php * app/Libs/UserSessionInterface.php * app/Libs/CoreUploadedFile.php * app/Libs/UserLoginInfo.php * app/Libs/Uploader/FileUploadConfig.php * app/Libs/Uploader/FileUploader.php	* app/Modules/User/Controllers/UserController.php (リクエストを受け取る処理を追加) * app/Modules/User/Views/user/index.html (リクエスト送信テスト用のフォームを追加)	
step10	抽象クラスを作り、コントローラーとモデルの初期化処理を実装します。 **ここで自作フレームワークはいったん完了とします。** 次のステップから、画面機能を実装します。	* app/Libs/Core/AbstractController.php * app/Libs/Core/AbstractModel.php * app/Libs/AbstractAdminController.php * app/Libs/AbstractUserController.php	* app/Modules/User/Controllers/UserController.php (AbstractUserControllerを継承する) * app/Modules/User/Controllers/IndexController.php (AbstractUserControllerを継承する)	
step11	ユーザ登録ページを実装します。	* app/Modules/Views/user/*html * app/Libs/Password.php * app/Models/UserModel.php * app/Repositories/UserRepository*.php	* app/Libs/Mailer/SwiftMailSender.php (メール送信処理を実装) * app/Modules/Controllers/UserController.php	* app/Repositories/Test*Repository*.php * app/Models/TestModel.php
step12	ログインページを実装します。	* app/Models/AuthModel.php * app/Modules/User/Controllers/AuthController.php * app/Modules/User/Views/auth/*.html		

step13	投稿ページを実装します。	* app/Models/ArticleModel.php * app/Modules/User/Controllers/PostController.php * app/Modules/User/Views/post/*.html * app/Modules/User/Controllers/ImageController.php * app/Libs/ImageManipulator.php * app/QueryServices/ArticleQueryService*.php * app/Repositories/Article*Repository*.php	* composer.json (intervention/imageを追加)	
step14	記事一覧ページを実装します。	* app/Modules/User/Controllers/ArticleController.php * app/Modules/User/Views/article/*.html * app/Libs/Pager.php		
step15	記事ランキングページを実装します。	* app/Console/SummaryRankingBatch.php * app/Libs/Core/ConsoleApplication.php * app/Modules/User/Views/article/rank.html		