# Use Cases

## Create Post for a Website

Description: An admin or system user can create a new post for a specific website.

* + Input: Website ID, post title, post description.
  + Output: Confirmation of post creation.

### Test Cases:

1. A computer screen shot of a computer program

   Description automatically generatedValidate that a post can be created with valid inputs.
2. A computer screen with text on it

   Description automatically generatedEnsure that a post cannot be created without a title or description.
3. Check that the post is linked to the correct website.

A screen shot of a computer program

Description automatically generated

1. A computer screen shot of a black screen

   Description automatically generatedVerify that duplicate posts are not created (e.g., the same title and description).

### Implementation:

1. Create a migration for the posts table.
2. Create a Post model.
3. Create a PostsController and define the store method.
4. Implement validation logic in the controller.
5. Use Eloquent to save the post in the database.

## Subscribe to a Website

Description: Users can subscribe to receive notifications for a specific website.

* + Input: Website ID, user email.
  + Output: Confirmation of subscription.

### Test Cases:

1. Validate that a user can successfully subscribe with a valid email.

A computer screen with text on it

Description automatically generated

1. Ensure that a user cannot subscribe without providing an email.

A screen shot of a computer program

Description automatically generated

1. Check that a user cannot subscribe more than once to the same website.

A computer screen with text and images

Description automatically generated

1. Validate that the subscription process is linked to the correct website.

### Implementation:

1. Create a migration for the subscriptions table.
2. Create a Subscription model.
3. Create a SubscriptionsController and define the store method.
4. Implement validation logic in the controller.
5. Use Eloquent to save the subscription in the database.

## Send Email Notifications

Description: When a new post is published, all subscribers to that website receive an email notification.

* + Input: Website ID, post title, post description.
  + Output: Email sent to all subscribers.

### Test Cases:

1. Verify that emails are sent to all subscribers upon post creation.
2. Ensure that emails contain the correct post title and description.
3. Check that no emails are sent for duplicate posts.
4. A computer screen shot of a program code

   Description automatically generatedTest that the email sending process works correctly with the background queue.

### Implementation:

1. Create a mailable class (e.g., YourPostCreatedMail).
2. In your PostsController, after saving the post, dispatch a job to send emails.
3. Use Laravel’s queue system to send emails in the background.

## Migrations for Required Tables

Description: Set up the database structure for posts and subscriptions.

* + Output: Database tables created.

### Test Cases:

1. Ensure that the migrations create the required tables (posts, subscriptions).
2. Validate that the columns in each table match the requirements.
3. Verify that the tables have appropriate indexes.

A computer screen shot of a program code

Description automatically generated

### Implementation:

1. Create the migrations using php artisan make:migration create\_posts\_table and php artisan make:migration create\_subscriptions\_table.
2. Define the structure of each table as shown above.

## Seed Data for Websites (Optional)

Description: Populate the database with predefined websites for testing.

* + Output: Initial data in the websites table.

### Test Cases:

1. Confirm that seeded data is correctly inserted into the database.
2. Ensure that the seeded data can be accessed through the API.

A black screen with white text

Description automatically generated

### Implementation:

1. Create a seeder (e.g., WebsiteSeeder).
2. Use factories to create seeded data.

## API Documentation (Optional)

Description: Provide documentation for the available APIs.

* + Output: Open API documentation or Postman collection.

### Test Cases:

1. Check that the documentation accurately reflects the available endpoints.
2. Ensure that the documentation includes examples for usage.

## Caching (Optional)

Description: Implement caching for frequently accessed data.

* + Output: Improved performance for API calls.

### Test Cases:

1. Validate that caching reduces the time taken for repeated API calls.
2. Ensure that the cache is updated correctly when data changes.

A computer screen with text on it

Description automatically generated

### Implementation:

1. Use Laravel’s Cache facade where applicable, e.g., in your WebsitesController.

## Events/Listeners (Optional)

Description: Use events to trigger actions (like sending emails) when posts are created.

* + Output: Responsive system that reacts to events.

Test Cases:

1. Verify that the event is triggered when a new post is created.
2. Check that the listeners correctly handle the event and perform the intended actions.

A screen shot of a computer program

Description automatically generated

### Implementation:

1. Create an event (e.g., PostCreated) and a listener (e.g., SendPostNotification).
2. Dispatch the event when a post is created.

Active Directory Components:

Tests\Feature:  
CreatePostTest.php

CreateSubscriptionTest.php

CreateWebsiteTest.php

Console:

Kernel.php

Console\Commands:

SendEmailsCommand.php

Providers:

App\Providers:  
AppServiceProvider

EventServiceProvider

Observers:  
App\Observers:  
WebsiteObserver.php  
PostObserver.php

SubscriptionObserver.php

Interactors:

App\Interactors:

CreateWebsiteInteractor.php

CreatePostInteractor.php

CreateSubscriptionInteractor.php

SendEmailInteractor,php

Requests Handlers:

App\Requests:

CreateWebsiteRequests.php

CreatePostRequests.php

CreateSubscriptionRequests.php

SendEmailRequest.php  
  
Validation Exceptions:

App\ValidationExceptions:

ValidationException.php

Controllers:

WebsiteController.php

PostController.php

SubscriptionController.php

App\Jobs:

SendPostEmails.php

SendWebsiteEmails.php

SendSubscriptionEmails.php

App\Mail:  
PostPublished.php

WebsiteCreated.php

UserSubscribed.php

Models:

Post.php

Subscription.php

Website.php

SentEmail.php

User.php

Database\Factories:

PostFactory,php

SubscriptionFactory.php

UserFactory.php

WebsiteFactory.php

SentEmailFactory.php

Database\Migrations:

create\_users\_table.php

create\_websites\_table.php

create\_posts\_table.php

create\_subscriptions\_table.php

add\_sent\_to\_posts\_table.php

create\_sent\_emails\_table.php

Database\Seeders:

DatabaseSeeder

PostSeeder

SubscriptionSeeder

WebsiteSeeder

Resources\Views:

createPost/index.blade.php

createWebsite/index.blade.php

createSubscription/index.blade.php

emails/postPublished.blade.php

emails/websiteCreated.blade.php

emails/userSubscribed.blade.php

Layout:

layouts/app.blade.php

Routes:

web.php

TO BE CREATED AND CODE: Directory Components:

Active Directory Components:

Tests\Feature:

SubscriberReceivesEmailTest.php

Console:

Kernel.php

Console\Commands:

SendEmailsCommand.php

Controllers:

WebsiteController.php

PostController.php

SubscriptionController.php

App\Jobs:

SendPostEmails.php

App\Mail:  
PostPublished.php

Models:

SentEmail.php

Database\Factories:

PostFactory,php

SubscriptionFactory.php

UserFactory.php

WebsiteFactory.php

Database\Migrations:

add\_sent\_to\_posts\_table.php

create\_sent\_emails\_table.php

Database\Seeders:

DatabaseSeeder

PostSeeder

SubscriptionSeeder

WebsiteSeeder

Resources\Views:

createPost/index.blade.php

createWebsite/index.blade.php

createSubscription/index.blade.php

emails/postPublished.blade.php

Layout:

layouts/app.blade.php

Routes:

web.php

Interactors:

App\src\Domain\Emails\Interactors:

SendEmailInteractor,php

Requests Handlers:

App\src\Domain\Emails\Interactors\Request:

SendEmailRequest.php

Resources\Views:

createPost/index.blade.php

createWebsite/index.blade.php

createSubscription/index.blade.php

emails/postPublished.blade.php

Layout:

layouts/app.blade.php

Routes:

web.php

while including following tests:

basic test

handling database

behavioral Tests

what are the test cases for usecases:

User Create Website

User Create Post

User Create Subscription

Subscribers Receive Emails

Use commands to send emails to the subscribers

Use queues to schedule sending in the background.

No duplicate posts should get sent to subscribers by email.

Whenever a new post is published on a particular website, all its subscribers shall receive an email with the post title and description in it. No authentication of any kind is required.