THE INITIAL DESCRIPTION OF THE OBJECT OF INTELLECTUAL PROPERTY
SPACEFEX – Compliance System Source Code

```
<?php

namespace SORQPAY\Modules\CaseManagement\Repositories;

use Illuminate\Contracts\Pagination\LengthAwarePaginator;
use Illuminate\Database\Eloquent\Collection;
use SORQPAY\Modules\CaseManagement\Exception\CaseNotFoundException;
use SORQPAY\Modules\CaseManagement\Models\Cases;
use SORQPAY\Modules\CaseManagement\Property\CaseType;
use SORQPAY\Modules\Users\Models\UserType\Agent;
use SORQPAY\Modules\Users\Models\UserType\AgentInterface;
use SORQPAY\Modules\Users\Models\UserType\UserInterface;
use SORQPAY\Modules\CaseManagement\Repositories\Filters\CaseBusinessProfileCompanyNameFilter;
use SORQPAY\System\Repositories\AbstractSorqpayRepository;
use SORQPAY\System\Repositories\Filters\ArrayFilter;
use SORQPAY\System\Repositories\Filters\LikeFilter;
use SORQPAY\System\Repositories\Filters\ExectFilter;
use SORQPAY\System\Repositories\Filters\ExectCustomerIdFilter;
use SORQPAY\System\Repositories\Filters\WhereInFilter;
use Spatie\QueryBuilder\AllowedFilter;
use Spatie\QueryBuilder\QueryBuilder;
use SORQPAY\Modules\Users\Models\Profile\BusinessProfile;
use SORQPAY\Modules\CaseManagement\Property\CaseStatus;
use Illuminate\Support\Facades\Log;
use Twilio\Rest\Chat\V1\Service\UserInstance;

/**
 * Class CaseRepository
 * @package SORQPAY\Modules\CaseManagement\Repositories
 *
 * @method Cases create(array $attributes)
 */
class CaseRepository extends AbstractSorqpayRepository
{
/**
 * @return string
 */
public function model(): string
{
        return Cases::class;
}

/**
 * @param string $id
 *
 * @throws CaseNotFoundException
 * @return Cases
 */
public function getById(string $id): Cases
{
        $case = $this->find($id);
        if (null === $case) {
                throw CaseNotFoundException::byId($id);
        }
```

```
        return $case;
}

/**
 * @param string $type
 * @return Collection
 */
public function getByType(string $type): Collection
{
        return $this->findAllBy('type', $type);
}

/**
 * @param UserInterface|AgentInterface $user
 *
 * @return LengthAwarePaginator
 */
public function getPaginatedByUser(UserInterface $user): LengthAwarePaginator
{
        $queryBuilder = QueryBuilder::for($this->model());
        if (true === $user instanceof Agent) {
                $queryBuilder
                        ->where(
                                [
                                        'type' => CaseType::INFORMATION_REQUEST,
                                        'business_profile_reference_code' => $user->getReferenceCode()
                                ]
                        )
                        ->orWhere(
                                [
                                        'assigned_by' => $user->getPersonalProfile()->getKey(),
                                        'assigned_to' => $user->getPersonalProfile()->getKey()
                                ]
                        );
        }

        // if case has incorrect id or manually deleted businessProfile
        // we will have an issue. we need to check that.
        $businessProfiles = BusinessProfile::query()->get();
        $queryBuilder->whereIn('business_profile_id', $businessProfiles->pluck('_id'));

        $queryBuilder->allowedFilters(

                AllowedFilter::custom('business_profile_id', new ExectFilter),
                AllowedFilter::custom('status', new WhereInFilter)->default(CaseStatus::$aemiFilterDefault),
                AllowedFilter::custom('title', new LikeFilter),
                AllowedFilter::custom('case_id', new LikeFilter),
                AllowedFilter::custom('tags', new ArrayFilter),
                AllowedFilter::custom('company_name', new CaseBusinessProfileCompanyNameFilter),
                AllowedFilter::custom('customer_id', new ExectCustomerIdFilter)
        );

        return $queryBuilder
                ->defaultSort('-created_at')
                ->allowedSorts(['created_at', 'deadline'])
                ->paginate();
}

/**
 * @param string $businessProfileId
 * @param string $status
 *
```

```
     * @return Collection
     */
    public function getCasesByBusinessProfileIdWhereStatusNot(string $businessProfileId, string $status): Collection
    {
        $this->applyConditions(
            [
                ['business_profile_id', '=', $businessProfileId],
                ['status', '!=', $status]
            ]
        );

        return $this->get();
    }

    /**
     * @param UserInterface|AgentInterface $user
     * @param string $type
     *
     * @return LengthAwarePaginator
     */
    public function getPaginatedByUserCaseType(UserInterface $user, string $type): LengthAwarePaginator
    {
        $queryBuilder = QueryBuilder::for($this->model());
        $queryBuilder
            ->where(
                [
                    'business_profile_id' => $user->getBusinessProfileId(),
                    'type' => $type
                ]
            );
        $queryBuilder->allowedFilters(
            AllowedFilter::custom('business_profile_id', new ExectFilter),
            AllowedFilter::custom('status', new ExectFilter),
            AllowedFilter::custom('title', new LikeFilter),
            AllowedFilter::custom('case_id', new LikeFilter),
            AllowedFilter::custom('tags', new ArrayFilter),
            AllowedFilter::custom('company_name', new CaseBusinessProfileCompanyNameFilter)
        );
        return $queryBuilder
            ->defaultSort('-deadline')
            ->allowedSorts('deadline')
            ->paginate();
    }

    public function getInfoCasesWithMessagesToUser(UserInterface $user)
    {

        return QueryBuilder::for($this->model())
            ->where(
                [
                    ['business_profile_id', $user->getBusinessProfileId()],
                    ['type', CaseType::INFORMATION_REQUEST],
                ]
            )
            ->orderBy('updated_at', 'desc')
            ->paginate();
    }
}

<?php

namespace SORQPAY\Modules\CaseManagement\Models;
```

```
use Carbon\Carbon;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\HasMany;
use Illuminate\Notifications\Notifiable;
use Illuminate\Support\Collection;
use Illuminate\Support\Str;
use Jenssegers\Mongodb\Eloquent\Model as Eloquent;
use ReflectionException;
use SORQPAY\Modules\CaseManagement\Event\CaseSaved;
use SORQPAY\Modules\CaseManagement\Property\CaseStatus;
use SORQPAY\Modules\CaseManagement\Property\CaseType;
use SORQPAY\Modules\FileManager\Models\Document;
use SORQPAY\Modules\FileManager\Property\DocumentType;
use SORQPAY\Modules\InformationRequest\Models\InformationRequest;
use SORQPAY\Modules\InformationRequest\Models\Messages;
use SORQPAY\Modules\Users\Exception\Profile\BusinessProfileNotFoundException;
use SORQPAY\Modules\Users\Exception\Profile\PersonalProfileNotFoundException;
use SORQPAY\Modules\Users\Models\Profile\BusinessProfile;
use SORQPAY\Modules\Users\Models\Profile\PersonalProfile;
use SORQPAY\System\Property\Connection;
use SORQPAY\System\Property\Table;
use Illuminate\Database\Eloquent\Collection as Eloquent_Collection;

/**
 * Class Cases
 * @package SORQPAY\Modules\CaseManagement\Models
 *
 * @property string $_id
 * @property string $business_profile_id
 * @property string $business_profile_reference_code
 * @property string $personal_profile_id
 * @property string $case_id
 * @property string $type
 * @property string $status
 * @property string $assigned_to
 * @property string $assigned_by
 * @property string $title
 * @property string $description
 * @property Carbon $deadline
 * @property Collection|array $tags
 * @property string|null $comply_advantage_search_id
 * @property Collection|array $comply_advantage_matches
 * @property array $payload
 * @property string $created_by
 * @property Carbon $created_at
 * @property Carbon $updated_at
 */
class Cases extends Eloquent
{
use Notifiable;

/**
 * @var string $connection
 */
protected $connection = Connection::MONGODB;
/**
 * @var string $collection
 */
protected $collection = Table::CASES;
/**
 * @var array $dispatchesEvents
 */
```

```
protected $dispatchesEvents = ['saved' => CaseSaved::class];
/**
 * @var array $attributes
 */
protected $attributes = ['status' => CaseStatus::NEW];
/**
 * @var array $dates
 */
protected $dates = ['deadline'];
/**
 * @var array $fillable
 */
protected $fillable = [
        'business_profile_id',
        'business_profile_reference_code',
        'personal_profile_id',
        'case_id',
        'type',
        'status',
        'assigned_by',
        'assigned_to',
        'title',
        'subject',
        'description',
        'deadline',
        'tags',
        'comply_advantage_search_id',
        'comply_advantage_matches',
        'payload',
        'created_by',
        'created_at',
        'updated_at'
];

protected $appends = ['messagesCount'];

public function getMessagesCountAttribute()
{
        return $this->messages()->count();
}

/**
 * @return Collection|CaseRelation[]
 */
public function getRelatedCases(): Collection
{
        return $this->hasMany(CaseRelation::class, 'belongs_to_case_id')->get();
}

/**
 * @return Collection|Document[]
 */
public function getDocuments(): Collection
{
        return $this->hasMany(Document::class, 'case_id')->where('type', DocumentType::CASE)->get();
}

/**
 * @throws BusinessProfileNotFoundException
 * @return BusinessProfile
 */
public function getBusinessProfile(): BusinessProfile
{
```

```
		/** @var BusinessProfile $businessProfile */
		$businessProfile = $this->belongsTo(BusinessProfile::class, 'business_profile_id')->first();
		if (null === $businessProfile) {
			throw BusinessProfileNotFoundException::byId($this->business_profile_id);
		}

		return $businessProfile;
	}

	/**
	 * @return PersonalProfile|null
	 */
	public function getPersonalProfile(): ?PersonalProfile
	{
		/** @var PersonalProfile|null $personalProfile */
		$personalProfile = $this->belongsTo(PersonalProfile::class, 'personal_profile_id')->first();
		return $personalProfile;
	}

	/**
	 * @param string $type
	 * @throws ReflectionException
	 */
	public function setTypeAttribute(string $type): void
	{
		$this->attributes['type'] = CaseType::create($type)->getValue();
	}

	/**
	 * @return PersonalProfile
	 */
	public function getAssignedBy(): PersonalProfile
	{
		/** @var PersonalProfile $personalProfile */
		$personalProfile = $this->belongsTo(PersonalProfile::class, 'assigned_by')->first();
		if (null === $personalProfile) {
			throw PersonalProfileNotFoundException::byId($this->assigned_by);
		}

		return $personalProfile;
	}

	/**
	 * @return null|PersonalProfile
	 * @throws \Exception
	 */
	public function getAssignedTo(): ?PersonalProfile
	{
		/** @var PersonalProfile $personalProfile */
		$personalProfile = $this->belongsTo(PersonalProfile::class, 'assigned_to')->first();

		//TODO: change to other exception type
		if ($this->assigned_to == null) {
			throw new \Exception('Unassigned');
		}

		if (null === $personalProfile) {
			throw PersonalProfileNotFoundException::byId($this->assigned_to);
		}

		return $personalProfile;
	}
```

```
/**
 * @param array $value
 */
public function setTagsAttribute(array $value = []): void
{
	$tags = Collection::make($value)->transform(
		function (string $tag): string {
			return Str::upper($tag);
		}
	);

	$this->attributes['tags'] = $tags->unique()->toArray();
}

/**
 * @return Collection
 */
public function getTagsAttribute(): Collection
{
	return Collection::make($this->attributes['tags'] ?? []);
}

/**
 * @return Collection
 */
public function getComplyAdvantageMatchesAttribute(): Collection
{
	return Collection::make($this->attributes['comply_advantage_matches'] ?? []);
}

/**
 * @return array
 */
public function getPayloadAttribute(): array
{
	return $this->attributes['payload'] ?? [];
}

/**
 * @throws PersonalProfileNotFoundException
 * @return PersonalProfile
 */
public function getCreator(): PersonalProfile
{
	/** @var PersonalProfile $personalProfile */
	$personalProfile = $this->belongsTo(PersonalProfile::class, 'created_by')->first();
	if (null === $personalProfile) {
		throw PersonalProfileNotFoundException::byId($this->created_by);
	}

	return $personalProfile;
}

public function getSubject(): ?string
{
	return $this->subject;
}

public function setSubject(?string $subject): self
{
	$this->subject = $subject;

	return $this;
```

```
}

public function requests(): HasMany
{
	return $this->hasMany(InformationRequest::class, 'request_id');
}

public function messages(): HasMany
{
	return $this->hasMany(Messages::class, 'case_id');
}

public function getId(): string
{
	return $this->_id;
}

public function getStatus(): string
{
	return $this->status;
}

public function setStatus(string $status): self
{
	$this->status = $status;

	return $this;
}

public function setTitle(string $title): self
{
	$this->title = $title;

	return $this;
}

public function getType(): string
{
	return $this->type;
}

public function getMessages(): Eloquent_Collection
{
```